PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
H04L 12/24, H04Q 11/04

A1

(11) International Publication Number: WO 98/42102

(43) International Publication Date: 24 September 1998 (24.09.98)

(21) International Application Number: PCT/CA98/00232

(22) International Filing Date: 16 March 1998 (16.03.98)

(30) Priority Data:

2,200,009	14 March 1997 (14.03.97)	CA
2,200,011	14 March 1997 (14.03.97)	CA
60/043,080	8 April 1997 (08.04.97)	US

(71) Applicant *(for all designated States except US)*: CROSSKEYS SYSTEMS CORPORATION [CA/CA]; 350 Terry Fox Drive, Kanata, Ontario K2K 2W5 (CA).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: FORGET, Leo [CA/CA]; 55 Forest Creek, Stittsville, Ontario K2S 1M1 (CA). CHRISTMAS, Mark [CA/CA]; 13 Riding Way, Kanata, Ontario K2M 1C3 (CA).

(74) Agent: MITCHELL, Richard, J.; Marks & Clerk, P.O. Box 957, Station B, Ottawa, Ontario K1P 5S7 (CA).

(81) Designated States: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, GM, GW, HU, ID, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published

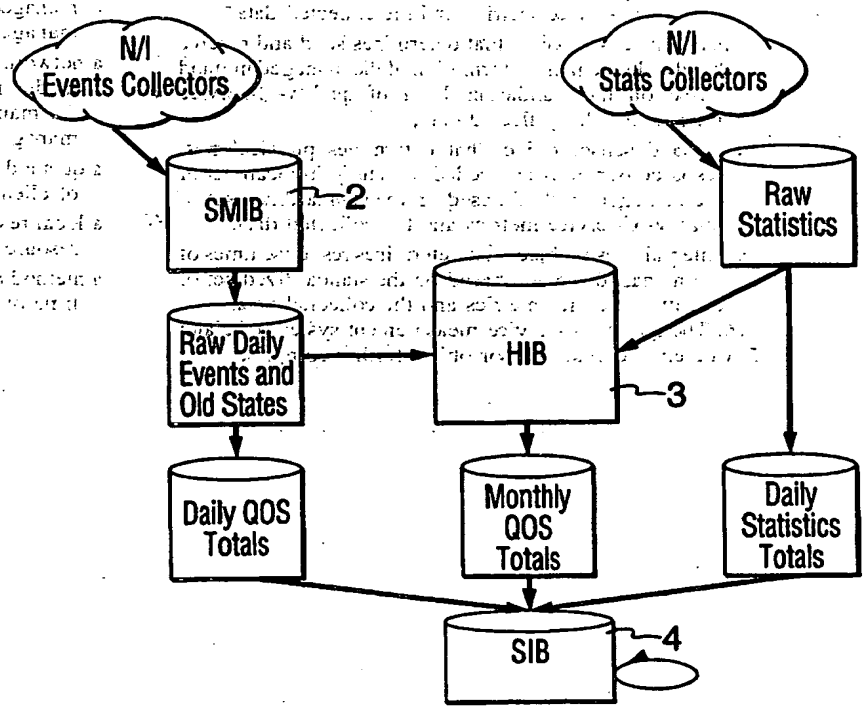
With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: SERVICE LEVEL AGREEMENT MANAGEMENT IN DATA NETWORKS

(57) Abstract

A method managing a telecommunications network which involves maintaining a database containing data relating to service level agreements with customers using an object model. Data from the network data relating to the performance of the network is continually compared with data stored in the database. A report is generated based on this data showing the performance levels for individual customers in meeting commitments stored in the database containing data relating to the service level agreements.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SERVICE LEVEL AGREEMENT MANAGEMENT IN DATA NETWORKS

The present invention relates to a method of managing a telecommunications network, and in particular to a method of monitoring the compliance with service level agreements.

More and more telecommunications services are now becoming available to the consumer. In packet switched networks, unlike circuit switched networks, customers are not given a dedicated circuit; their data is statistically multiplexed with data from other sources. Each customer pays for a particular level of service, and it is therefore important to ensure that the customer is receiving the level of service he has paid for.

There is a thus need for a system that manages service level agreements (SLAs) between telecommunications service providers and their business customers. Part of the management process that relates to SLAs is the comparison of the service providers' performance vis-à-vis specific guarantees that it may provide to its customer. Such a system must be capable of handling vast amounts of data.

A object of the invention is to provide such a system.

According to the present invention there is provided a method managing a telecommunications network comprising the steps of maintaining a database containing data relating to service level agreements with customers using an object model, receiving from the network data relating to the performance of the network, comparing the data received from the network with data stored in said database, and generating a report based on said data showing the performance levels for individual customers in meeting commitments stored in said database containing data relating to service level agreements.

The method may be implemented, for example, on a Sun Sparc Ultra 2 Unix-based workstation and, for example, work in conjunction with a Newbridge Networks Corporation 46020 network manager.

In a preferred embodiment the event is generated when the discrepancy between performance levels and commitments exceed a predetermined threshold value.

As a specific example, consider a case where the service provider guarantees a Cell Loss Ratio of a specific percentage for an ATM (Asynchronous Transfer Mode) PVC service. This ratio is guaranteed over a monthly period. A PVC (permanent virtual circuit) is a logical circuit between two points used to carry bi-directional information. The Cell Loss Ratio indicates the quality of a specific service by providing a measurement of the amount of data loss by the service provider's network due to various reasons such as network congestion and network failure. The Cell Loss Ratio for an ATM PVC based service is calculated using the following formula:

$$CLR = ((\sum Ra + \sum Rz) - (\sum Ta + \sum Tz))/(\sum Ra + \sum Rz) * 100$$

where CLR = Cell Loss Ratio

Ra = number of data cells received by the provider's network from side A

Rz = number of data cells received by the provider's network from side Z

Ta = number of data cells transmitted by the service provider to side A

Tz = number of data cells transmitted by the service provider to side Z.

Data which relates to Ra, Rz, Ta, and Tz is typically available from the ATM switching equipment in the form of statistical counters that are regenerated every fifteen minutes. Various other statistics, provided by the ATM switching equipment, are required for the verification of other service quality metrics.

These statistical counters must be collected by a management system in order to aggregate and summarize the various quality metrics associated with each service. This involves storing and managing millions of statistical counters that are required to manage the services offered by a typical service provider. In order to gain a better perspective as to the magnitude of this need, let's consider the following typical example.

A service provider must measure service performance metrics on 50,000 PVC services. Each PVC service generates 2 statistical reports per 15 minute interval (1 for each side of the PVC). Each statistical report consists of an identifier, a time stamp, and 8 statistical counters. Thus in this scenario, the management system must process 9.6 million records (50,000*2*96 intervals). Furthermore, raw statistical information must typically be available on-line for up to 60 days. Thus the system must manage (9.6 * 60) 576 million records or 54 gigabytes of storage space.

Ideally, all of this information should be in one database table. Given the size of the table (576 million records), it must be indexed or searches in this table would be cumbersome and time consuming. Conversely, the processing to add 9.6 million records to an indexed table of 576 million records would take days, as the amount of time required to load data into an indexed table grows exponentially with respect to the volume of data already in the table.

The difficulty is how to manage effectively and efficiently this vast quantity of data that changes rapidly in real time. This is quite a daunting task.

In a preferred embodiment, a plurality of working table fragments forming part of a fragmented table are created in memory, data is loaded in successive predetermined time periods into successive table fragments in a predetermined sequence, and the data are processed separately when loaded into the table fragments.

The data are preferably loaded into said table fragments using a round robin table fragmentation strategy.

The invention also provides a telecommunications network service level manager comprising a database containing data relating to service level agreements with customers using an object model; means for receiving from the network data relating to the

performance of the network; means for continually comparing the data received from the network with data stored in said database, and means for generating a report based on said data showing the performance levels for individual customers in meeting commitments stored in said database containing data relating to said service level agreements.

The invention still further provides a method of controlling a computer in an object-oriented environment wherein descriptors implemented as an object oriented class are used to store meta information on other classes in the system.

The invention will now be described in more detail, by way of example only, with reference to the accompanying drawings, in which:

Figure 1 is a software architecture diagram of a system in accordance with the invention;

Figure 2 is a data flow diagram;

Figure 3 is a database entity relationship diagram for the object model;

Figure 4 shows the entities contained in the SMIB;

Figure 5 shows the entities contained in the HIB;

Figure 6 shows the entities contained in the SIB;

Figure 7 is a software Architecture Diagram showing daily data process flow;

Figure 8 shows the system table fragmentation strategy;

Figure 9 shows a fragmented HIB table and related entries in the DBSpace usage table in the SMIB;

Figure 10 illustrates the identifying and detaching aged data in a HIB table;

Figure 11 illustrates the attaching of new data to a table in a HIB and updating the Dbspeace usage table in the SMIB;

Figure 12 shows combinations of start and completion times of states, caused by events in the SMIB;

Figure 13 shows the object models for a telcom information management architecture;

Figure 14 illustrates the different types of descriptors that are employed in the system;

Figure 15 shows the top layer object model;

Figure 16 shows the admin entity in Figure 13; and

Figure 17 shows the service layer object model of Figure 13.

As will be apparent from the following description, the invention implements an object model to efficiently construct the management system capable of handling a large volume of information. Referring now to Figure 1, Database Monitor (ckdbmon) 1 exchanges messages with the system databases, namely the service management information database (SMIB) 2, the historical information database (HIB) 3, summarized information database (SIB) 4, and Network Interface Systems Director (the Keep Alive Process) 5. The ckdbmon 1 does not interface directly with the system databases, but with a relational

database management system (RDBMS) 7 employed by the system, namely Informix Online version 7.13.

The Monthly Frame Relay and ATM Statistics are derived from the Daily Frame Relay and ATM summarizations already contained in the SIB 4. This is indicated in Figure 2 with the asterisk (*). Figure 3 illustrates the process flow of this data as it relates to the daily processing of the system Data.

The Data Management Framework consists of a database monitoring tool, load and unload utilities, and several scripts that employ the load and unload utilities in order to migrate and summarize data between the various databases. The monitoring component utilizes the Network Interface keep alive process and the monitoring tool's output is logged in a Network Interface Logging Tool compatible format 8. The advantages of doing so are that the logging interface is common between the database and network interface frameworks, and the ability to reuse coded and tested tools is provided. The load and unload utilities also use the log tool format to post all operational and alert messages, as do the utility scripts.

The role of the monitoring tool 1 is to ensure that the system databases do not exceed predefined space utilization thresholds, that the System databases remain active and available to the end users, and that the Informix Online specific event log file (typically called online.log) does not grow too large. Should sections of the System databases become too full (exceeding the threshold), a message is posted, via the log tool, to the System Administrator (not shown).

If, for some reason, the System databases are in an unavailable state (due to Informix Online being brought off-line), the monitor 1 will make several attempts to restart Informix Online and will again post an alert to the System Administrator stating that Informix Online is off-line and it (the monitor) is attempting to restart it.

When the Informix specific event log file exceeds the predefined size that the monitor is gauging it against, the monitor 1 will remove log file entries (checkpoint notification messages only), starting with the oldest ones, until the log file again fits within a specified size range.

These monitoring functions provide the System Administrator with more freedom, as less manual checking of the System Databases status is necessary. Additionally, the monitor promotes greater System database availability as the most common database operation-stopping difficulty, namely running out of space, is monitored and alerts are sent in anticipation of a problem occurring, not just in response to one.

The Director 5 (the Network Interface keep alive process) ensures that the database monitor (ckdbmon) 1 is started and remains active. The ckdbmon, in turn, ensures that the database management system 7 (Informix Online) remains active. Ckdbmon 1 will

shutdown gracefully, if it receives a shutdown command from the Director 5. However ckdbmon will not shut down Informix Online when the Director's shutdown message is received.

Informix may only be brought off-line by the Administrator explicitly issuing the valid Informix commands (*onmode -uy*; followed by *onmode -ky*). Informix is immune to the Director's shutdown commands, and cannot be brought off-line by ckdbmon, so that the System databases may remain on-line, and available, even if the Director or ckdbmon should experience difficulties and shutdown.

Figure 3 illustrates an example of an object model in accordance with the invention. In Figure 3, the object entities are defined as follows.

Service Entities

Customer

The customer is a legally identified organization that is contracting for the supply of one or more services from one or more service providers.

Parent to: Contract, Current Service, Historical Service, Contact via Customer id number.

Contract

The contract is a legal administrative and technical document describing what will be provided to the Customer, how and when it will be provided and the terms and conditions under which it will be provided. It also describes the obligations placed upon the Customer.

Parent to: Current Service, Historical Service, Contact, Contract Threshold via Contract id number.

Child of: Customer via Customer id number.

Contract Threshold

Contract Thresholds are SLA thresholds that are associated with a Contract.

Child of: Contract via Contract id number.

Current Service

A service is anything that the service provider determines that customers wish to purchase and that the service provider is willing to supply. A Current Service record contains information on a currently provided service.

Parent to: Service Component, Contact via Service id number.

Child of: Customer via Customer id number, Contract via Contract id number, Service Profile via Service Profile id number.

Historical Service

A service is anything that the service provider determines that customers wish to purchase and that the service provider is willing to supply. A Historical Service record contains information that was current until the associated customer, contract, or service profile changed.

Child of: Customer via Customer id number, Contract via Contract id number, Service Profile via Service Profile id number.

Contact

A Contact unambiguously identifies a person who carries out a role associated with a specific service entity (Customer, Contract, Current Service).

Child of: Customer, Contract, Current Service via service entity id number, Person via Person id number.

Service Component

A service component is any network entity that is used within a service offered to a customer. Each service can consist of one or more service components. At the time of the network entity creation, the service component will not be related to any service. It may be assigned to a service at a later time.

Parent to: Frame Relay PVC, Frame Relay CTP, ATM PVC, ATM CTP, TDM Circuit, TDM NI, Service Information, Events, Current State, Old State via Service Component id number.

Child of: Current Service via Service id number.

Service Profile

A service profile describes the characteristics of a specific service or group of services.

Parent to: Current Service, Historical Service, Valid Service Component, Service Profile Threshold via Service Profile id number.

Valid Service Components

A description of each Valid Service Component that can be associated with a Service Profile.

Child of: Service Profile via Service Profile id number.

Service Profile Threshold

Service Profile Thresholds are SLA thresholds that are associated with a Service Profile.

Child of: Service Profile via Service Profile id number.

Administrative Entities

Role

A role identifies all the actions, with respect to the System, that a user in a specific job function is permitted to perform.

Parent to: User, Entity Access via Role id number.

User

A user is a communicating entity which is registered in the Resolve Databases for the purpose of performing tasks with the Resolve System.

Parent to: Audit Entry via User id number. Additionally, all creation of and modifications to Customers, Contracts, Current Services, Historical Services, Service Profiles, Network Entities (PVC's, CTP's, NI, etc.), Roles, Persons can be performed by existing User ids only.

Child of: Person via Person id number, Role via Role id number.

Person

A person is a specific individual. All Contacts must be Persons. All Users must be Persons.

Parent to: Contact, User via Person id number.

Audit Entry

An audit entry is generated each time user performs an operation. This entry contains references to the user, the action performed, the type of entity operated on, and the time the operation occurred.

Child of: User via User id number, Valid Entity Operations via Entity/Operation number.

Entity Access Control

This entity is used to define what operations, on what entity, are permitted for a given role. One instance of this record is created for each entity to which a specific role has privileges on.

Child of: Role via Role id number, Valid Entity Operations via Entity/Operation number.

Valid Entity Operations

This entity is used to define valid operation and entity combinations for the Resolve System. These combinations are then used to assign Entity Access to Roles, and to create Audit information.

Parent to: Audit Entry, Entity Access Control via Entity/Operation number.

Network Entities

FR PVC

Frame Relay Permanent Virtual Circuit.

Parent to: FR NP, FR CTP via Network Entity id number.

Child of: Service Component via Network Entity id number.

FR CTP

Frame Relay Circuit Termination Point. A FR PVC will have two or more (two in this release) Termination Points. Future services, such as multicast connections will have multiple CTP's.

Child of: FR PVC, Service Component via Circuit id number.

FR NP

Frame Relay Network Performance. This entity consists of network performance statistics collected from each FR PVC path end.

Parent to: FR PVC Daily NP, FR PVC Monthly NP via FR PVC id number, and Path End id number.

Child of: FR PVC via FR PVC id number.

FR PVC Daily NP

FR PVC Daily Network Performance. Daily summarization of FR PVC Network Performance.

Child of: FR NP via FR PVC id number, and Path End id number.

FR PVC Monthly NP

FR Monthly Network Performance. Monthly summarization of FR PVC Network Performance.

Child of: FR NP via FR PVC id number, and Path End id number.

ATM PVC

Asynchronous Transfer Mode Permanent Virtual Circuit.

Parent to: ATM NP, ATM CTP via Network Entity id number.

Child of: Service Component via Network Entity id number.

ATM CTP

Asynchronous Transfer Mode Circuit Termination Point. An ATM PVC will have two or more (two in this release) Termination Points. Future services, such as multicast connections will have multiple CTP's.

Child of: ATM PVC, Service Component via Circuit id number.

ATM NP

Asynchronous Transfer Mode Network Performance. This entity consists of network performance statistics collected from each ATM PVC path end.

Parent to: ATM PVC Daily NP, ATM PVC Monthly NP via ATM PVC id number, and Path End id number.

Child of: ATM PVC via ATM PVC id number,

ATM PVC Daily NP

ATM PVC Daily Network Performance. Daily summarization of ATM PVC Network Performance.

Child of: ATM NP via ATM PVC id number, and Path End id number.

ATM PVC Monthly NP

ATM Monthly Network Performance. Monthly summarization of ATM PVC Network Performance.

Child of: FR NP via FR PVC id number, and Path End id number.

TDM Circuit

Time Division Multiplexing Circuit.

Parent to: TDM NI via Network Entity id number.

Child of: Service Component via Network Entity id number.

TDM NI

Time Division Multiplexing Network Interface.

Child of: TDM Circuit, Service Component via TDM Circuit id number.

Event

An Event is information describing an occurrence on the network entity for which a report is required.

Parent to: Current State, Old State via Event id number.

Child of: Service Component via Network Entity id number.

Current State

The Current State of each Service Component is described. The Event id number links this entity to the Event which caused the Service Component to be in its Current State.

Child of: Service Component via Network Entity id number, Event via Event id number.

Old State

Previous states of each Network Entity. This entity also describes the duration of time (in seconds) that the Network Entity was in a particular state.

Child of: Service Component via Network Entity id number, Event via Event id number.

FR PVC Daily QOS

FR PVC Daily Quality of Service. This entity describes the quality of service provided, for each Frame Relay PVC Network Entity, with respect to availability time, outage time, etc. on a daily basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

FR PVC Monthly QOS

FR PVC Monthly Quality of Service. This entity describes the quality of service provided, for each Frame Relay PVC Network Entity, with respect to availability time, outage time, etc. on a monthly basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

ATM PVC Daily QOS

ATM PVC Daily Quality of Service. This entity describes the quality of service provided, for each Asynchronous Transfer Mode PVC Network Entity, with respect to availability time, outage time, etc. on a daily basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

ATM PVC Monthly QOS

ATM PVC Monthly Quality of Service. This entity describes the quality of service provided, for each Asynchronous Transfer Mode PVC Network Entity, with respect to availability time, outage time, etc. on a monthly basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

TDM Daily QOS

TDM Daily Quality of Service. This entity describes the quality of service provided, for each Time Division Multiplexing Circuit Network Entity, with respect to availability time, outage time, etc. on a daily basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

TDM Monthly QOS

TDM Monthly Quality of Service. This entity describes the quality of service provided, for each Time Division Multiplexing Circuit Network Entity, with respect to availability time, outage time, etc. on a monthly basis. The QOS statistics are derived from the data contained in the Old State entity.

Child of: Old State via Network Entity id number.

System Entities

Service Info

This entity is used as an attach point for addition description information relating to the Network Entities.

Child of: Service Component via Network Entity id number, Info Type via Info Type id number.

Info Type

This entity is used to specify the type of service information that can be associated with each Network Entity.

Parent to: Service Info via Info Type id number.

Stat Collector Info

This entity contains information regarding each set of statistics that is collected.

Event Collector Info

This entity contains information regarding each event collection session.

46020 Event Translation

This entity maps 46020 events to Resolve events.

46020 Stat Translation

This entity maps 46020 statistics to Resolve statistics and indicates which statistics should be gathered for the Resolve Databases.

46020 CallAtt Translation

This entity maps 46020 objects to Resolve Network Entities. This includes the ability to map more than one Network Management System's objects.

Table Version Info

This entity is used to track the version of each physical table in the Resolve Databases. This table ensures that incorrect versions of data are not restored.

Archive Info

This entity tracks all Resolve archives, both full database backups of the SMIB and SIB, and daily table backups within the HIB.

DBSpace Usage

This entity keeps track of all the dbspaces available and in use in the Resolve Databases. This entity is used to maintain the large inflow and outflow of data to the HIB.

The Physical Database Design is the physical, or actual, representation of the Object Model and Logical Database Design. In most instances, the physical design maps quite closely to the logical design, but some deviations may be to achieve greater response performance, or to take advantage of additional features of the RDBMS employed, or to accommodate a lack of required features in the RDBMS.

The SMIB 2 is an operational data store. It contains both 'soft data' - data (customer, contract, SLA) that can be derived from other Service Provider systems, and data that is in a constant state of flux - Service and Service Component data.

The SMIB 2 is the definitive source from which to derive inventory and status reports on the Networks, the impact on Service Provider Customers, and the appropriate individuals to contact with respect to Network events.

Due to the fact that a sizable portion of the SMIB's data is changed daily, the SMIB is enabled as a transaction logging database. That is, any changes made to the SMIB are not only stored in the database, but also recorded in transaction logs that can be replayed in the event that disaster recovery is necessitated, thus the SMIB can be recovered up to its most recent update

Note that the data contained in the Events, Current State, and Old State entities is only a single days worth. This data is migrated, nightly, to the HIB. The SMIB is shown in Figure 4.

The HIB 3, shown in Figure 5, is a very large store of data. It contains Network Events, the corresponding Network Entity states, and the Network Performance Statistics for all the Network Entities that are currently being tracked (as indicated in the SMIB). By volume of data, the HIB is approximately 40 to 50 times larger.

In the simplest sense, the HIB is a data warehouse. It contains very large volumes of data, covering the same Network Entities over a period of time (60 days, in the case of Resolve 1.0), and the data is never updated by end users, or by connecting systems.

The HIB is NOT a data warehouse from the view that it does not contain data brought together from multiple heterogeneous data sources, but this is a discussion that is of little relevance to this document. Suffice to say, that the HIB contains a very large volume of data that is quite static in nature.

The daily Events, Current States, and Old States are migrated to the HIB from the SMIB nightly, and the Network Performance Statistics are loaded from flat files (created by the Network Interface Stats Collectors - see *Resolve Release 1.0 "Architect" Network Interface for 46020 Detailed Functional Specification*, reference [8]) into the HIB nightly.

The data in the HIB is held on-line (within the active database) for a period of 60 days, and is then purged. It is, however, saved on tape, and may be recovered for additional analysis with the assistance of the Resolve Administrator.

Unlike the SMIB 2, the HIB 3 does not employ transaction logging, meaning that the HIB cannot be recovered to the most recent point in time. Recovery to the most recent point in time, however, is not necessary as the HIB does not permit user updates against it. Since the only updates are performed by nightly processes, the new data added to the HIB daily is archived to tape by one of these processes. Thus, any disaster recovery may be performed by the Resolve Administrator using the daily data that has been archived to tape.

The SIB 4 contains the end product of all the data collection and processing efforts. It is here that the end users of Resolve 1.0 may most easily extract meaningful information. All the information in the SIB is summarized and processed data extracted from the HIB. The processes to create SIB information may be customized to suit a particular Service Provider.

The information in the SIB, like that of the HIB, is static in nature as it is not updated or modified by users or processes. Because of its condensed nature (a single days worth of statistics for one service component equates to 96 records in the HIB, but only one record in the SIB), the SIB can present information covering a broader time period (180 days). The Quality of Service entities are derived from Old State data in the HIB, and the Network Performance Statistics are summarizations of Network Performance Statistics in the HIB.

Like the HIB 3, the SIB 4 does not employ transaction logging. It does however, have regular backups of the entire SIB made. In the event of a disaster, the Resolve Administrator could restore the SIB back to its current state by restoring the most recent backup.

The SIB 4, shown in Figure 6, contains the end product of all the data collection and processing efforts. It is here that the end users of may most easily extract meaningful information. All the information in the SIB is summarized and processed data extracted from the HIB. The processes to create SIB information may be customized to suit a particular Service Provider.

The information in the SIB, like that of the HIB, is static in nature as it is not updated or modified by users or processes. Because of its condensed nature (a single days worth of statistics for one service component equates to 96 records in the HIB, but only one record in the SIB), the SIB can present information covering a broader time period (180 days). The Quality of Service entities are derived from Old State data in the HIB, and the Network Performance Statistics are summarizations of Network Performance Statistics in the HIB.

Like the HIB, the SIB does not employ transaction logging. It does however, have regular backups of the entire SIB made. In the event of a disaster, the Resolve Administrator could restore the SIB back to its current state by restoring the most recent backup.

The operation of the system will now be described. When the database server is started (or restarted), the instance of the Director on the server starts a ckdbmon process for each instance of Informix Online that exists on that server. That is, if two Informix Online servers are running on the same workstation (this is a distinct possibility), two instances of ckdbmon will be started. Each instance of ckdbmon will start its own instance of the ckdbmon log tool 8 (ckdblog).

Additionally, the cron table is set so that regularly scheduled database jobs are initiated. These jobs are run nightly and accomplish the task of migrating data from the SMIB 2 to the HIB 3, and summarizing the data in the HIB 3 and deriving quality of service (QoS) information, and statistical summaries, and placing this information in the SIB 4. Other

nightly jobs archive the data collected daily for the HIB 3 and store it for future reference in an AIB (Archive Information Base). Figure 2-3 describes the nightly flow of data and the high level processes involved in this flow of data. Lastly, utilities to remove old data from the HIB and SIB, and to restore old, archived data from the AIB to the HIB 3 and SIB 4 are provided.

With this feature network statistical data that can reach volumes of up to 600 million rows in a single table at one time. Obviously, a table of such proportions would require some indexing or searching for specific data within it would be a tiresome and time consuming task. Conversely, attempting to load an additional 1 million records into this table, having an index, could take days, as the amount of time required to load data into an indexed table grows exponentially with respect to the volume of data already in the table.

In order to avoid this costly operation, a new work table is created, with the same structure as the table containing all the existing data, but without any indexes. Data is loaded at a much faster rate (the amount of time required to load data into a non-indexed table is a linear function, with the constant being very close to 1 - that is the number of records by a set factor of time). Figure 4 illustrates the use of table fragmentation as it is employed by the System Database Management Framework.

Data is loaded, from an ASCII delimited flat file into the temporary working table, that has been created in its own dbspace (1). One days worth of data is loaded into each dbspace. Once this work table has been loaded, advantage can be taken of its smaller size, relative to the larger HIB data table (1 million rows vs. 600 million rows for stats), and the fact that it contains a single full days worth of data (a day is the smallest unit that the SIB data is summarized on), to perform any summarizations or quality of service (QoS) derivations to be loaded into the SIB on this work table.

In addition, the original ASCII delimited file is archived in the AIB (2), and this fact recorded in the SMIB, so that this data can be re-examined even after it has been aged and purged from the HIB and SIB. Again, this ASCII file contains one full days worth of data.

The aged data that currently exists in the HIB can be easily removed by simply detaching the dbspace that contains that particular days worth of data from the main table and then deleting that single portion (5). This reduces the time that would be required if the data were to be deleted directly from the main table (the dbspace that has just been cleared of aged data can then be reused as the work dbspace).

Finally, the work table is attached to the main table, and the indexes on the main table are rebuilt - but only the portions of the index relating to the newly attached data.

Table fragmentation, that is the actual data in a fragmented table can be separated, using one of three methods:

1. Expression Based Fragmentation - all data is placed in a dbspace based on a expression involving a column or columns within the table (e.g. a table containing phone numbers may be fragmented using expressions based on ranges of area codes).
2. Hash Fragmentation - a unique, or key column in the fragmented table is put through a hashing formula to determine which dbspace that particular record should reside in.
3. Round Robin - all data that is inserted into the fragmented table is distributed evenly across all dbspaces in the fragmented table.

An interesting characteristic about the Round Robin method is that any data that is in a dbspace that is attached to a fragmented table (vs. being inserted directly into the fragmented table), is not redistributed to other dbspaces within the fragmented table. This means that as long as the System utilities do not insert data directly into the main, fragmented tables in the HIB, but instead, create work tables and load them and attach them to the main fragmented tables, it is possible to know exactly which dbspace contains a particular days worth of data.

While the same statement is true when using the other two fragmentation methods, both of those require the Informix engine to check each row of data in both the fragmented main table and the work table prior to allowing the attach to take place. Conversely, no row checking is done when performing an attach operation using the Round Robin method. The difference between Round Robin and Expression Based fragmentation, in the time required to perform an attach operation, is huge. Therefore the Data Management Framework uses Round Robin fragmentation, and the rule that no data may be directly inserted into a main table in the HIB is strictly enforced.

Each of the main data tables in the HIB (events, old states, FR Network Performance, and ATM Network Performance) are fragmented using Round Robin fragmentation, and each has a finite number of dbspaces dedicated to it. Typically, there are 60 dbspaces per table, for 60 days worth of data, and an additional 15 dbspaces for the retrieval of historical data that has already been removed from the HIB and now resides only in the AIB.

Since the row size and the number of records per day varies between the tables in the HIB, the dbspace sizes are different for each table. This necessitates that the each dbspace is dedicated to one and only one table.

Because the system HIB only stores a finite number of days worth of data, the dbspaces for each table can be recycled, with old aged data being removed and new data being added using the same dbspace. Figures 7 - 9 demonstrate how this is performed from a dbspace usage level.

The table called dbspace usage is a table in the SMIB (the actual name is txd_dbspaceusage). This table is used by the utilities loadhib_event, loadhib_ostate, loadhib_frnp, and loadhib_atmnp, and it allows these utilities to identify the dbspace that contains data that is a particular age (in the example, 60 days old), to remove that data by detaching the dbspace, and then load the new data into that dbspace, and update the dbspace usage table (Figure 3-4).

The SMIB is the operational data store of the system Databases. This means that the data contained in it is timely and can change frequently.

In order to keep track of current network path (or network entity) status, the operational and administrative states of each path, as well as the events that caused a path to be in that state, are recorded in the SMIB. Event data is stored in the tev_event table, while the current state of each path is stored in the tcs_currstate table. Querying the current state table will allow the user to build up-to-the-minute inventory reports of available paths, and operational reports of path status'.

Since this state information is also required to derive long term availability information (used to measure compliance to service level agreements), the non-current, or 'old' state data is also saved and maintained (this is stored in the tos_oldstate table).

Within the SMIB, each time the state of a path is updated (in the current state table), a database trigger (add_state_rec) creates a new record in the old state table indicating the old states of the path, the time that it originally entered that state, and the amount of time (in seconds) it remained in that state.

All of this data (the events and states) is migrated nightly to the HIB as it quickly changes from being critical, timely information, to data that can be used to track SLA compliance.

Since events are constantly occurring, and it is necessary to keep the events, current state, and old state tables synchronized, a suspend_collection signal is sent to the events collectors, via the Network Interface Director's interface process called RCI. This suspend_collection signal will cause the events collectors to stop inserting new events into the SMIB and instead, store those events in buffers until the previous days worth of events and old state records can be removed from the SMIB. At that point a resume_collection signal is sent, via RCI to the Director and, to the events collectors, and normal processing resumes, with any buffered events being processed first.

Figure 11 is a simple timeline demonstrating the various combinations of events and states (with respect to start and completion times) that may exist in the events table in the SMIB. Events 0 and 1 occurred prior to the day that is being processed (October 10, 1996), but the state that events 0 and 1 placed paths 0 and 1 in, respectively, continued into the processed day. Event 2 placed path 2 in a state that both started and completed (at

Event 2') within the processing day. Event 3 occurred in the processing day, but the state of path 3, caused by event 3, continued into the next (the current) day, as did the state that event 0 placed path 0 in.

Since the SIB reports on summarization's in strict one day intervals, and since the HIB contains all data that is one-day old or older, from Figure 8, it is apparent that a method to include those events that are still current (the state has not changed yet), but that occurred prior to the current day must be devised.

The strategy, as implemented in the System Database script unload_smib, is to add a column onto the end of each old state record in the old state tables in both the SMIB and HIB. This column, called the partial column, contains a 0 if the record has been completed prior to the end of the previous processing day (midnight) - the states caused by events 1 and 2, in Figure 8, fall into this category if processing for October 10, 1996 is being performed (as expected) in the early morning of October 11, 1996.

Any events in which the state is still current, as is the case with states 0-0' and 3-3', in Figure 11 (as derived by joining the events table with the current state table) will have that current state placed in the old state table, but be indicated as partial by placing a value of 1 in the partial column of the old state table. The current state records (relating to events 0 and 3) will remain in the SMIB, as will the matching event records (events 0 and 3), but a record of the event that caused the current state, and the duration, up until midnight, is now recorded and moved to the HIB. Since states that do not complete prior to midnight processing are placed in the HIB as partial records, these records must be replaced with either a new partial records or completed records the next processing period. Thus, full data, up until 12:00am of the current day is available in the HIB, and data integrity is insured as the partial records are updated or replaced. This is shown in the tables below.

Events

PathId	EventId	time	EvType
0	0	t0	A
1	1	t1	A
2	pre2	tpre2	B
3	3	t3	A
2	2	t2	A
1	1'	t1'	B
2	2'	t2'	B

Current State

PathId	State	time	EventId
0	A	t0	0
3	A	t3	3
1	B	t1'	1'
2	B	t2'	2'

Old State

PathId	EventId	time	State	Duration	Partial
3	pre3	tpre3	B	t3-tpre3	0
2	pre2	tpre2	B	t2-tpre2	0
1	1	t1	A	t1'-t1	0
2	2	t2	A	t2'-t2	0

Contents of the Events, Current State, and Old State tables in the SMIB, at 11:59pm, October 10, 1996

The Events table will contain all events that occurred either during the hours of the processing day - Oct. 10 (events 2, 3, 1', 2') or those which were current events as of 12:00am of the processing day (events 0, 1). Note that the old state table will only contain records of previous states that were in effect at some time during the processing day. This

is why there are old state records relating to events pre2 and pre3 - the states pre2-2 and pre3-3 were in effect as of 12:00am, Oct. 10.

The following table illustrates the changes that occur in the old state table in the SMIB during nightly processing, as performed by unload_smib.sh. The current states of the paths are written to the old state table as partial records with duration's calculated to midnight of the processing day.

Old State

PathId	EventId	time	State	Duration	Partial
3	pre3	tpre3	B	t3-tpre3	0
2	pre2	tpre2	B	t2-tpre2	0
0	0	t0	A	midnight Oct.10 - t0	1
1	1	t1	A	t1'-t1	0
3	3	t3	A	midnight Oct.10 - t3	1
2	2	t2	A	t2'-t2	0
1	1'	t1'	B	midnight Oct.10 - t1'	1
2	2'	t2'	B	midnight Oct.10 - t2'	1

Partial Records placed in the SMIB Old State table during unload_smib.sh processing

All old state and event records which have a timestamp of a day prior to the current day - Oct. 11 (states 0-0', 1-1', 2-2', and 3-3', and events 0, 1, 2, 3, 1', and 2') will be copied into ASCII delimited files for loading into the HIB, and all non-current event (events 0, 1, 2, and 3) and all old state records (states 0-midnight Oct. 10, 1-1', 2-2', 3-midnight Oct. 10) will be deleted from the SMIB. The following tables illustrate the state of the HIB, prior to nightly processing of Oct. 10th data, and the state of the SMIB and HIB after processing of Oct. 10th data.

Events

PathId	EventId	time	EvType
0	0	t0	A
1	1	t1	A
2	pre2	tpre2	B
3	pre3	tpre3	B

Old State

PathId	EventId	time	State	Duration	Partial
0	0	t0	A	midnight Oct.9 - t0	1
1	1	t1	A	midnight Oct.9 - t1	1
2	pre2	tpre2	B	midnight Oct.9 - tpre2	1
3	pre3	tpre3	B	midnight Oct.9 - tpre3	1

The state of the HIB Events and Old State tables prior to unload_smib.sh processing for Oct. 10th data

Prior to loading the event and old state records into the HIB, any event records which are related to partial old state records, and the partial old state records themselves (in the event and old state tables in the HIB), are deleted. These will be replaced by either new partial records, or completed records from the most recent day prior to the current day.

Events

PathId	EventId	time	EvType
0	0	t0	A
3	3	t3	A
1	1'	t1'	B
2	2'	t2'	B

Current State

PathId	State	time	EventId
0	A	t0	0
3	A	t3	3
2	B	t2'	2'

Old State

PathId	EventId	time	State	Duration	Partial
no data					

SMIB data in the Events, Current State, and Old State tables immediately after unload_smib.sh processing of Oct. 10th data

Events

PathId	EventId	time	EvType
0	0	t0	A
1	1	t1	A
3	3	t3	A
2	2	t2	A
1	1'	t1'	B
2	2'	t2'	B

Old State

PathId	EventId	time	State	Duration	Partial
3	pre3	tpre3	B	t3-tpre3	0
2	pre2	tpre2	B	t2-tpre2	0
0	0	t0	A	midnight Oct.10 - t0	1
1	1	t1	A	t1'-t1	0
3	3	t3	A	midnight Oct.10 - t3	1
2	2	t2	A	t2'-t2	0
1	1'	t1'	B	midnight Oct.10 - t1'	1
2	2'	t2'	B	midnight Oct.10 - t2'	1

HIB data in the Events and Old State tables immediately after unload_smib.sh and loadhib_event.sh and loadhib_ostate.sh processing of Oct. 10th data

There is one additional occurrence that must be processed in the Events and Old State tables in the SMIB. It is when there are new events, that have occurred after midnight of the processing day, but prior to the nightly processing being performed. The old state and event records of this type must be kept after the nightly processing, or the next day's processing will be inaccurate. To accommodate this occurrence, a partial type of 2 is placed on these records. They are not unloaded, nor are they deleted from the SMIB. The last step of the nightly unload_smib process sets these partial types back to 0, after the other event and old state records have been unloaded and deleted from the SMIB.

There exists, within the System Database Management Framework, the ability to restore aged data that has previously been removed, or purged, from the HIB. The principle behind this function is similar to the principle applied by the HIB storage management - tracking dbspaces availability.

Each table type in the HIB (currently Events, Old States, FR Network Performance, and ATM Network Performance) has a limited number (typically 15) spare dbspaces reserved for aged data. This is data that is older that what is considered active data in the HIB (older than 60 days in this example).

The following table illustrates how the data in the dbspace usage table is applied to locate a dbspace of type event that is not currently used and is available (indicated by a NULL value in the last usage date column, and the 1 in the available column).

DBSpace Usage Table

DBSpaceName	DBSpace Number	Object Type	Last Usage Date	Available
db1	6	event	1996-10-09	0
db2	10	event	1996-10-08	0
db3	14	event	1996-10-07	0
db4	18	event	1996-10-06	0
db60	262	event	1996-08-10	0
db61	266	event		1
db62	270	event		1

dbspace usage table - available dbspace highlighted

The actual aged data ASCII delimited files are tracked by another storage management table in the SMIB call the archive information table. This table gets updated every time an archive of HIB data occurs. The type of data (events, old state, etc), the volume label of the tape or file that the ASCII file is stored on, the date of the data, and the version of the table structure (for future use) is recorded.

When a user of the System System wishes to analyze aged data, a request for the data is made to the System Administrator. The System Administrator can then use the restore script to first, select the particular data that was requested, then restore it to the HIB. When that data is restored, the dbspace usage table is updated to indicate that this data is now in the HIB and the dbspace used is not available for other restores until this data has been removed again from the HIB.

Event Table

db1	db2	db3		db58	db59	db60	db61	db62		db75

DBSpace Usage Table

DBSpaceName	DBSpace Number	Object Type	Last Usage Date	Available
db1	6	event	1996-10-09	0
db2	10	event	1996-10-08	0
db3	14	event	1996-10-07	0
db4	18	event	1996-10-06	0
db60	262	event	1996-08-10	0
db61	266	event	1996-06-01	0
db62	270	event		1

updated dbspace usage table with restored aged data

Newly aged data (data that has just become older than the active data) is automatically purged with each nightly run of the loadhib scripts. However, aged data that has been restored by the restore is not purged. This is due to the fact that the restored data was restored for further analysis and should therefore be kept in the HIB until it is no longer required. The removal of this data requires proactive steps to be taken by the System Administrator. Using the purge_hib and purge_sib scripts the System Administrator can list all data that is older than active data in each of the HIB and SIB respectively.

purge_hib will use the dbspace usage table to determine the number of days old the aged HIB data is relative to the current day. Once purge_hib is executed, any data as old or older than the number of days old selected will be purged and the dbspace usage table will again be set in indicate that the dbspaces that were used for the purged data are now available, and the last usage date is again set to NULL.

purge_sib will scan the date values in the SIB to build of list of days of aged data. This is a slower process, but due to the lessor volume of data in the SIB (relative to the HIB)

performance should not be an issue. As with purge_hib, purge_sib will delete any data that is as old or older than the days old value passed to it.

Informix Online provides support for Referential Integrity in the form of Referential Constraints (foreign keys). The RI constraints ensure that a child record cannot be added if the relating value does not exist in the parent table, and conversely, that a parent record cannot be deleted if a child record relating to it exists. These relations deal with physical occurrences of records.

However, some entities within the System databases are related to each other on a logical level. That is, some entities (user, person, service contract, customer, etc) can be flagged as *logically* deleted, in which case they are no longer available for reference or manipulation via the Configuration GUI or Reporting tools, but they *physically* remain in the database. Since these records are not physically removed, there is no way of enforcing RI constraint rules. In response to this, what has been implemented is a set of triggers, activated upon the update of the 'deleted' indicator column in those tables, that call stored procedures that check for child records that have not been logically deleted. Table 3-1 lists each parent table, its child tables, and the trigger that is activated upon a logical delete.

Parent table	Child table	Trigger
tcu_customer	tcr_contract	process_cust_upd
	tco_contact	
tcr_contract	tse_currservice	process_contr_upd
	tco_contact	
tsp_servprofile	tse_currservice	process_sp_upd
tse_currservice	tsc_servcomp	process_serv_upd
	tco_contact	
tpe_person	tco_contact	process_pers_upd
tro_role	tus_user	check_role_child
	tea_entityaccess	

Data Integrity Triggers of the SMIB

Due to the very large numbers of DASD devices required for a full System, there are several files that must be customized on a per installation basis. The altering of these files can have dire consequences as they relate to the System.

1. links.sh - this file creates the symbolic links for the database that are used by Informix Online for dbspace files. This file will require that the devices linked to, and the chunks linked to each device are defined. Additionally, the symbolic links to the tape

devices required by the System Databases (two devices - one for log files, the other for archives) must be defined.

2. onspace.sh - this file creates the dbspaces on top of the symbolic links created by links.sh. The onspace commands specify not only the dbspace name and the path of the chunk supporting the dbspace, but also the bytes offset and the size of the dbspace. For this reason (specifically the offsets), onspace.sh must match the links.sh file. If it does not, the installation will proceed, but the System System will not function as some dbspaces will not be brought on-line due to incorrect space allocation (incorrect offsets).

3. dbspaceusage.sql - this file calls the stored procedure, init_dbspaceusage, which seeds the dbspace usage table for storage management. If the system you are installing will store 60 days of active HIB data, this file does not need to be changed. However, if you plan on storing more days or less days of HIB data, the first parameter in each execute procedure statement must be updated to reflect this.

4. storage.cfg - this file indicates where files are located, names of system tables, and the number of days of HIB and SIB data to be stored. If the number of days of HIB data stored is not 60, or the number of days of SIB data to be stored is not 180, this file must be updated to reflect this.

When the System Database is first installed, it is prepared, and expects, to accept data immediately. Conversely, if data is not submitted to the HIB processes for a period of time immediately after the system has been installed, The Data Management Framework processes will automatically perform the necessary alterations to ensure smooth operation.

Descriptors are used in the system as more fully described with reference to Figures 13 to 16.

Figure 13 is an overview of the object models for a TIM (Telecom Information Management) architecture.

Figure 14 shows the types of descriptors that are provided in a typical system for managing service level agreements. There is a top level descriptor, and derived descriptors relating to various aspects of the system.

The top level descriptor stores meta information on entities. The base class provides a template where unique Ids, names and descriptions are stored. The derived classes define additional qualities for specific descriptors.

In a service level management system, the use of descriptors enables new service level agreement thresholds to be added to the system without modifying the service profile code. For example:

Export Control: Public

Cardinality n

Hierarchy:

Superclasses None

Private Interface:

Attributes:

type : short

The descriptor uniquely identifies the descriptor record attached to a specific class

Name:char[200]

The descriptor name is the unique name of a descriptor

Description :char[200]

The description is a brief summary of the purpose of the descriptor

State Machine: No

Concurrency Sequential

Persistence Persistent

In Figure 15, the classes in the top layer object model of Figure 14 are defined as follows.

Class name:

Descriptor

Documentation:

This class is used to store meta information on entities. The base class provides a template where unique IDs, names and descriptions are stored. Derived classes are used to define additional qualities for specific descriptors. The intent behind the Descriptor concept is to minimize the impact of adding new functionality on critical parts of the system. For example, new SLA thresholds can be added to the system without modifying the Service Profile code.

Superclasses:

<none>

Roles/Associations:

<none>

Attributes:

type : short

The descriptorId uniquely identifies the descriptor record attached to a specific class.

name : char[80]

The descriptorName is the unique name of a descriptor.

description : char[200]

Figure 16 also illustrates how descriptors are used to dynamically add new capabilities to the system. As new entities or operations on entities are defined and implemented, new entity operation descriptors are defined and added. When these descriptors are added, the user management module becomes aware of these capabilities. The user management module is used to give/deny access to specific parts of the system.

Each instance of a descriptor sources its information from a specific row in a relational database table. The following table illustrates the instance data associated with the service component descriptors for use in a system for managing service level agreements in a telecommunications network.

Service Component Descriptor

Type	Name	Description
0	Undefined	Undefined Service Component.
1	TDM Path	A logical end to end connection implemented using time division multiplex (TDM) technology.
2	FR PVC	A frame relay path. A permanent virtual end-to-end connection implemented with frame relay technology.
3	ATM VCC	A virtual channel connection. A collection of connections that form an end-to-end path through a network
4	ATM VPC	A virtual path connection (VPC). A logical communication channel that is available across the physical ATM interface and that can carry one or more virtual channels.
5	TDM NI	A TDM Network Interface
6	FR CTP	Frame Relay Circuit Termination Point.
7	ATM CTP	ATM Circuit Termination Point.

If, for example, the service provider receives a requirement to support ATM UNI services, The following new descriptors are added.

Entity → ATM UNI

Service Component → ATM UNI

New descriptors are added for SLA thresholds, and statistics.

New valid operations instances are created for ATM UNI. No software is required to change in the user / security management module.

New service profiles can be created using ATM UNI without any modifications to the software in this critical area.

New services can be configured with ATM UNI service components assigned to them.

Within the configurator, the only software modifications required are new screens to view details associated with the ATM UNI.

The service definition layer is shown in Figure 17.

Class name:

Descriptor

Documentation:

This class is used to store meta information on entities. The base class provides a template where unique IDs, names and descriptions are stored. Derived classes are used to define additional qualities for specific descriptors. The intent behind the Descriptor concept is to minimize the impact of adding new functionality on critical parts of the system. For example, new SLA thresholds can be added to the system without modifying the Service Profile code.

Superclasses:

<none>

Roles/Associations:

<none>

Attributes:

type : short

The descriptorId uniquely identifies the descriptor record attached to a specific class.

name : char[80]

The descriptorName is the unique name of a descriptor.

description : char[200]

The description is a brief summary of the purpose of the descriptor.

Has-A Relationships:

<none>

Operations:

<none>

Class name:

ServiceEntity

Documentation:

Service entities are the base components of service model, representing the customer, the contracts owned by that customer, the services contained within the contract, and the service profiles desribing the services.

Superclasses:

Entity

Roles/Associations:

responsibleFor in association Contact_ServiceEntity

collectedBy in association ServiceEntity_UserLogEntry

Attributes:

whoCreated : integer

The ID of the user who created the entity.

whoLastModified : integer

The ID of the user who last modified the entity. If the deleted flag is set this attribute holds the id of the user who deleted the entity.

Has-A Relationships:

<none>

Operations:

Delete( )

The deletion of a service entity will be tracked as a modification with the result of the delete flag being set.

Class name:

SLAThresholdDescriptor

Documentation:

A template for a Threshold object that defines the characteristics and description but not the specific value of the threshold. An SLA threshold descriptor must be created for each new threshold type that the system will support.

Superclasses:

Descriptor

Roles/Associations:

canMeasure in association ServCompDesc_SLAThreshDesc

measures in association ContractThresh_SLAThresh

mapsTo in association SLAThresh_ServProfThresh

Attributes:

defaultValue : double

A default value for the threshold. The value can be customized in each instance of the SLAThreshold.

serviceCompDescType : short

Foreign key to the type of service component this threshold can apply to.

units : char[40]

Description of the unit of measurement.

relatedClass : short

Indicates whether this is a contract related threshold, or a service profile related threshold.

Has-A Relationships:

<none>

Operations:

<none>

Class name:

FunctionDescriptor

Documentation:

This class is used to enumerate the descriptions of functions or job roles that can be associated with a contact person within the system. A Function descriptor must be created for each new function that the system will support.

Superclasses:

Descriptor

Roles/Associations:

performs in association Contact_FunctionDesc

Attributes:

<none>

Has-A Relationships:

<none>

Operations:

<none>

Class name:

Contact

Documentation:

A Contact unambiguously identifies a person who carries out a role associated with a specific Service Entity. The class provides the necessary information to contact that person. * * definition from NM Forum - SMART Performance Reporting White Paper, September, 1995 (NMF/SPT95-15)

Superclasses:

<none>

Roles/Associations:

hasContact in association Contact_ServiceEntity

performedBy in association Contact_FunctionDesc

actsAs in association Person_Contact

Attributes:

serviceEntityId : integer

Foreign key to the service entity class. References the service entity this contact can be used for.

serviceEntityType : short
Foreign key to service entity table.
personId : integer
Foreign key to the person class. References the person who acts as the contact.
personType : short
Foreign key to person table. This field always contains the same value indicating the entity type is "PERSON".
functionDescType : short
Foreign key to the Fucntion Descriptor class.
Has-A Relationships:
<none>
Operations:
<none>
Class name:
ContractThreshold
Documentation:
Contract Thresholds are SLA Thresholds that are associated with a Contract. The SLA is a set of technical, administrative and management parameters that the service provider can report against. They typically are based on objective measures and have a high correlation with the users' perception of the quality of service. Each parameter is typically a threshold that, when surpassed, means that the quality test in question has failed. * * definition from NM Forum - SMART Performance Reporting White Paper, September, 1995 (NMF/SPT95-15
Superclasses:
<none>
Roles/Associations:
usesThreshold in association ContractThreshold_Contract
measuredAgainst in association ContractThresh_SLAThresh
Attributes:
contractId : integer
A foreign key to the entity ID of the associated contract.
contractType : short
Foreign key to contract table. This field always contains the same value indicating the entity type is "CONTRACT".
thresholdType : short
Foreign key to the SLA Threshold Descriptor table.
val : double

The numeric value of the threshold.

Has-A Relationships:

<none>

Operations:

<none>

Class name:

Customer

Documentation:

The Customer is a legally identified organization that is contracting for the supply of one or more services from one or more service providers. * * definition from NM Forum - SMART Performance Reporting White Paper, September, 1995 (NMF/SPT95-15)

Superclasses:

ServiceEntity

Roles/Associations:

belongsTo in association Customer_Contract

ownedBy in association Service_Customer

Attributes:

name : char[80]

The name of the customer or company.

idNumber : char[20]

A unique number identifying the customer and assigned by operational staff (i.e. customizable).

comments : char[255]

Any comments pertaining to a specific customer may be added to this field.

address : char[255]

A street address for the customer.

Has-A Relationships:

<none>

Operations:

Delete( )

A customer cannot be deleted while it is associated with a contract or service. All related contact information is deleted when a customer is deleted.

Class name:

Contract

Documentation:

The Contract is a legal administrative and technical document describing what will be provided to the Customer, how and when it will be provided and the terms and condition

under which it will be provided. It also describes the obligations placed upon the Customer. * * definition from NM Forum - SMART Performance Reporting White Paper, September, 1995 (NMF/SPT95-15)

Superclasses:

ServiceEntity

Roles/Associations:

maintains in association Customer_Contract

isContainedBy in association Contract_Service

usedIn in association ContractThreshold_Contract

Attributes:

customerId : integer

Foreign key pointing to the entity ID of the customer.

customerType : short

Foreign key to customer table. This field always contains the same value indicating the entity type is "CUSTOMER".

effectiveDate : time

The date on which the contract came into effect.

number : char[20]

A number used to track the contract against external systems. This number should be unique.

comments : char[255]

Any comments specific to the contract.

expiryDate : time

The date on which the contract expires.

Has-A Relationships:

<none>

Operations:

Delete( )

A contract cannot be deleted while services are associated with it. All contact information associated with the contract is deleted when the contract is deleted.

Create( )

The relation to a customer is set at creation time and cannot be changed subsequently.

Class name:

Service

Documentation:

A Service is anything that the service provider determines that Customers wish to purchase and that the service provider is willing to supply. * More specifically (within the

context of Resolve), a Service is a telecommunications product sold to a Customer by a service provider which is managed by the service provider. Services include transmission facilities and associated applications. A Service can consist of multiple Service Components. * definition from NM Forum - SMART Performance Reporting White Paper, September, 1995 (NMF/SPT95-15)

Superclasses:

ServiceEntity

Roles/Associations:

contains in association Contract_Service

owns in association Service_Customer

describes in association ServiceProfile_Service

containedBy in association ServiceComponent_Service

Attributes:

customerId : integer

Foreign key to the customer class. References the customer who owns this service.

customerType : short

Foreign key to customer table. This field always contains the same value indicating the entity type is "CUSTOMER".

contractId : integer

Foreign key pointing to the entity ID of the contract.

contractType : short

Foreign key to contract table. This field always contains the same value indicating the entity type is "CONTRACT".

serviceProfileId : integer

Foreign key to Service Profile class.

serviceProfileType : short

Foreign key to service profile table. This field always contains the same value indicating the entity type is "SERVICE PROFILE".

inServiceDate : time

The date that the service came into effect.

serviceName : char[80]

A label for this service.

comments : char[255]

Any information added by the user which is specific to this service.

Has-A Relationships:

<none>

Operations:

Delete( )

Deletion of this class follows the rules of deletion for the entity class, and leads to deletion of all related contact information.

Class name:

CurrentService

Documentation:

This class is used to track current services. Current services become historical services everytime a modification is made to their associations with a customer, contract or service profile record.

Superclasses:

Service

Roles/Associations:

<none>

Attributes:

<none>

Has-A Relationships:

<none>

Operations:

<none>

Class name:

HistoricalService

Documentation:

This class is used to track services that are no longer in use. This information is necessary for historical reporting. Everytime the associated customer, contract or service profile for a current service is changed, a historical service is created.

Superclasses:

Service

Roles/Associations:

<none>

Attributes:

expiryDate : time

Date the service was modified or went out of use.

Has-A Relationships:

<none>

Operations:

<none>

Class name:

Person

Documentation:

This class contains pertinent information on a specific individual.

Superclasses:

AdminEntity

Roles/Associations:

hasProperties in association Person_Contact

hasProperties in association Person_User

Attributes:

name : char[20]

Full name of the person.

position : char[40]

Short description of position held within the given organization.

organization : char[80]

Name of the organization with which this person is associated.

telephoneNumber : char[20]

Telephone number associated with the person.

faxNumber : char[20]

Fax number associated with the person.

emailAddress : char[40]

Electronic mail address for the person.

mailingAddress : char[80]

Full mailing address for the person.

comments : char[255]

Additional comments specific to the person.

Has-A Relationships:

<none>

Operations:

<none>

The described system provides an efficient method of managing service level management agreements in a packet switched network that is capable of handling vast quantities of data in a flexible manner.

Claims:

1. A method managing a telecommunications network comprising the steps of:

a) maintaining a database means containing data relating to service level agreements with customers using an object model,

b) receiving from the network data relating to the performance of the network,

c) continually comparing the data received from the network with data stored in said database, and

d) generating a report based on said data showing the performance levels for individual customers in meeting commitments stored in said database containing data relating to said service level agreements.

2. A method as claimed in claim 1, wherein said event is generated when the discrepancy between performance levels and commitments exceed a predetermined threshold value.

3. A method as claimed in claim 1 or 3, wherein a plurality of working table fragments forming part of a fragmented table are created in memory, data are loaded in successive predetermined time periods into successive table fragments in a predetermined sequence, and the data are processed separately when loaded into the table fragments.

4. A method as claimed in claim 4, wherein the data loaded using a round robin technique.

5. A method as claimed in any one of claims 1 to 4, wherein descriptors implemented as an object oriented class are used to store meta information on other classes in the system.

6. A method as claimed in any one of claims 1 to 4, wherein a base descriptor class provides a template where unique identifiers, names and descriptions are stored.

7. A method as claimed in claim6, wherein derived classes are used to define additional qualities for specific descriptors.

8. A method as claimed in claim 7, wherein derived classes are implemented for service entities, functions, statistics, Service Level Agreement thresholds, operations, service components, etc...

9. A telecommunications network service level manager comprising:

a) database means containing data relating to service level agreements with customers using an object model;

b) means for receiving from the network data relating to the performance of the network;

c) means for continually comparing the data received from the network with data stored in said database, and

d) means for generating a report based on said data showing the performance

levels for individual customers in meeting commitments stored in said database containing data relating to said service level agreements.

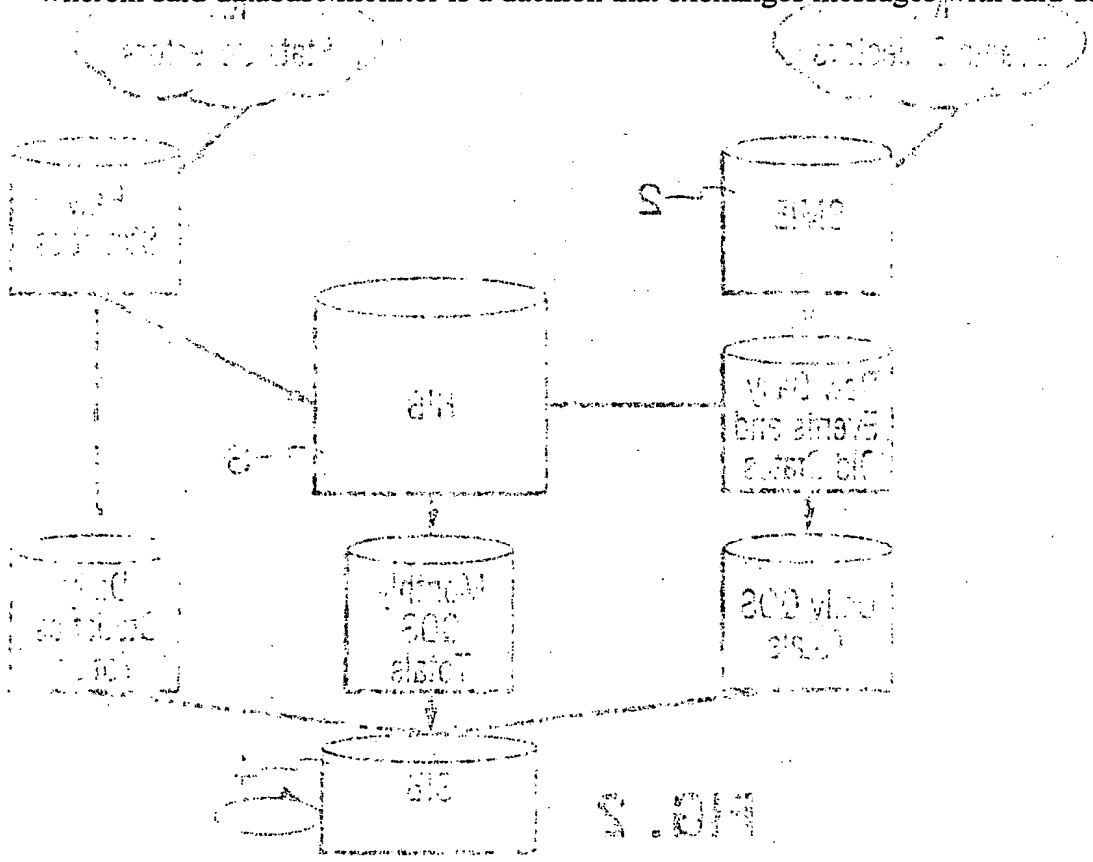
10. A telecommunications network service level manager as claimed in claim 9, wherein said database comprises a plurality of working table fragments forming part of a fragmented table are created in memory, means are provided for loading data in successive predetermined time periods into successive table fragments in a predetermined sequence, and means are provided for processing the data separately when loaded into the table fragments.

11. A telecommunications network service level manager claims 9 or 10, further comprising means for implementing descriptors as an object oriented class are used to store meta information on other classes in the system.

12. A telecommunications network service level manager as claimed in claim 11, wherein a base-descriptor class provides a template where unique identifiers, names and descriptions are stored.

13. A telecommunications network service level manager as claimed in claim 11, wherein said database means comprises a service level management database (SMIB), a historical information database (HIB), and summarized information database (SIB) under the control of a database monitor.

14. A telecommunications network service level manager as claimed in claim 13, wherein said database monitor is a daemon that exchanges messages with said databases.



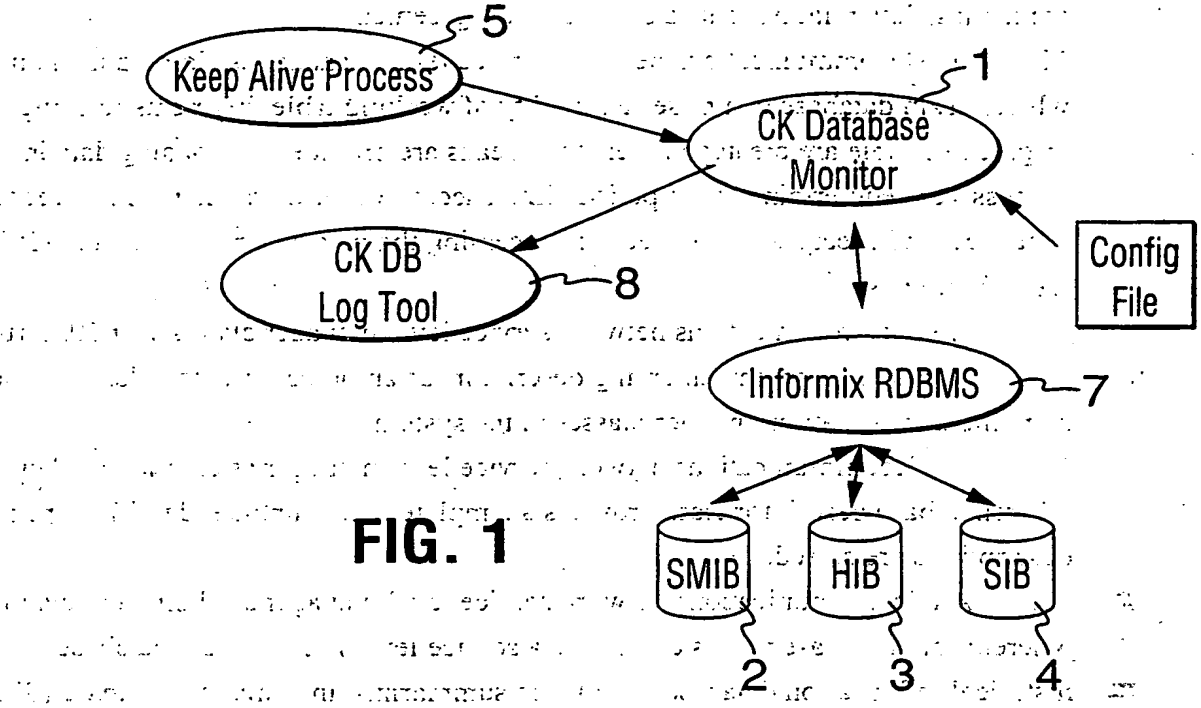


FIG. 1

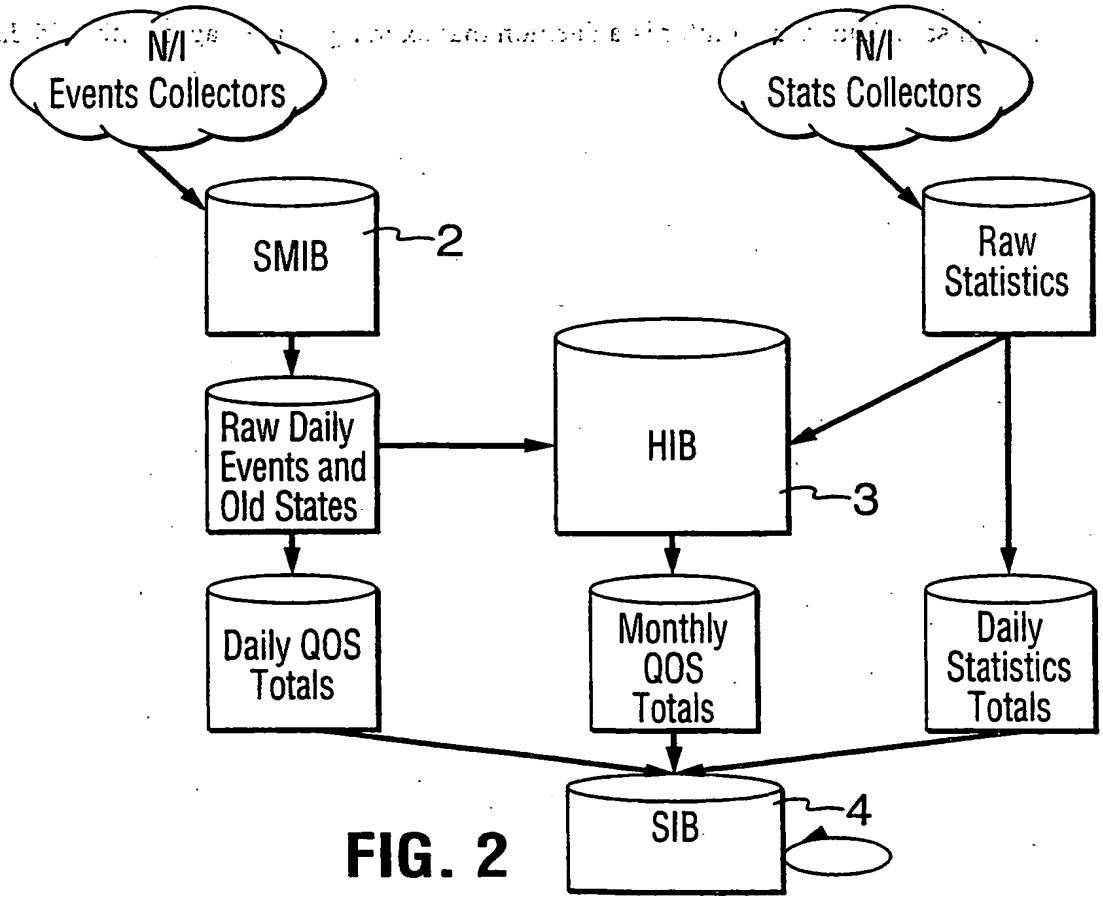


FIG. 2

FIG. 3

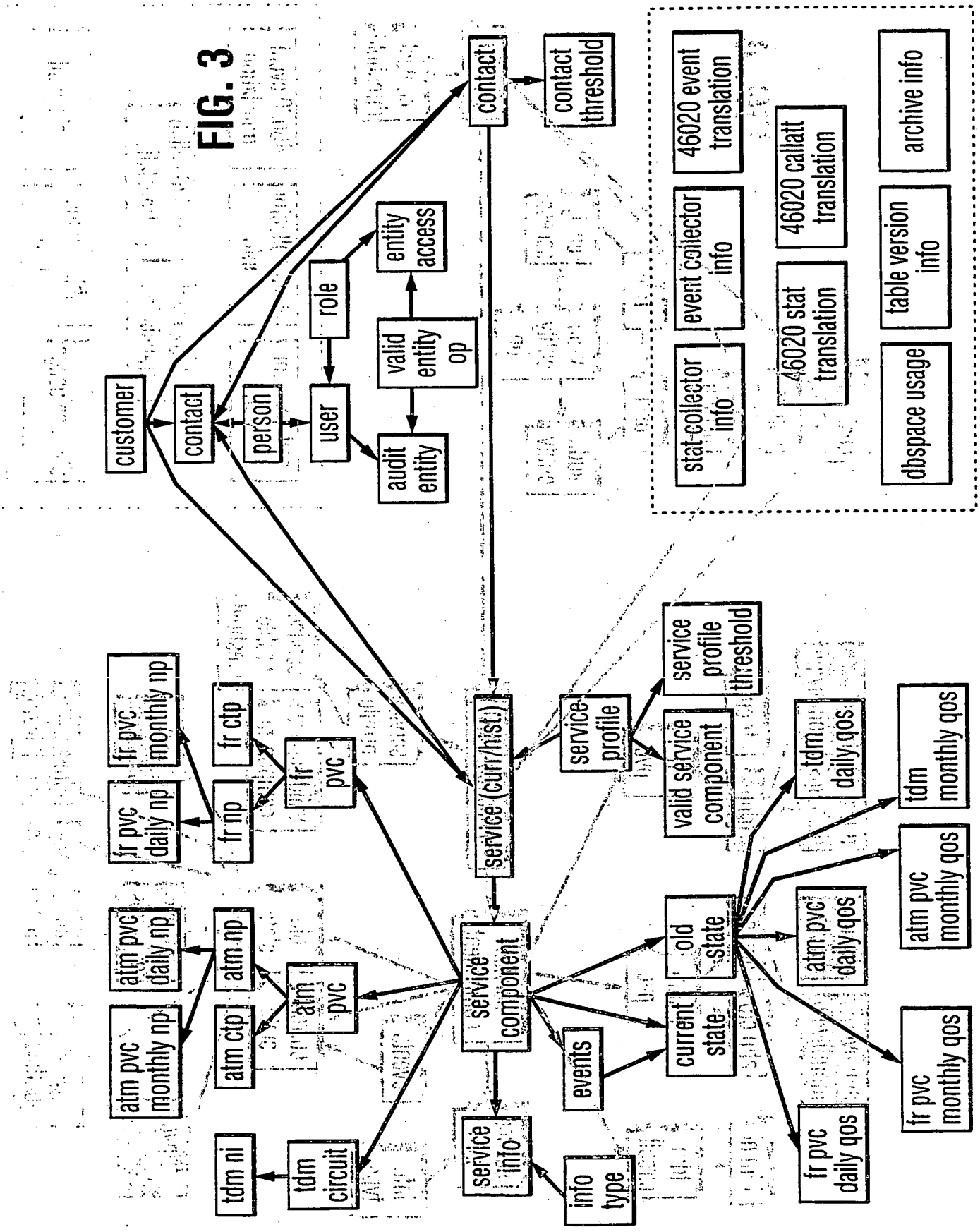


FIG. 4

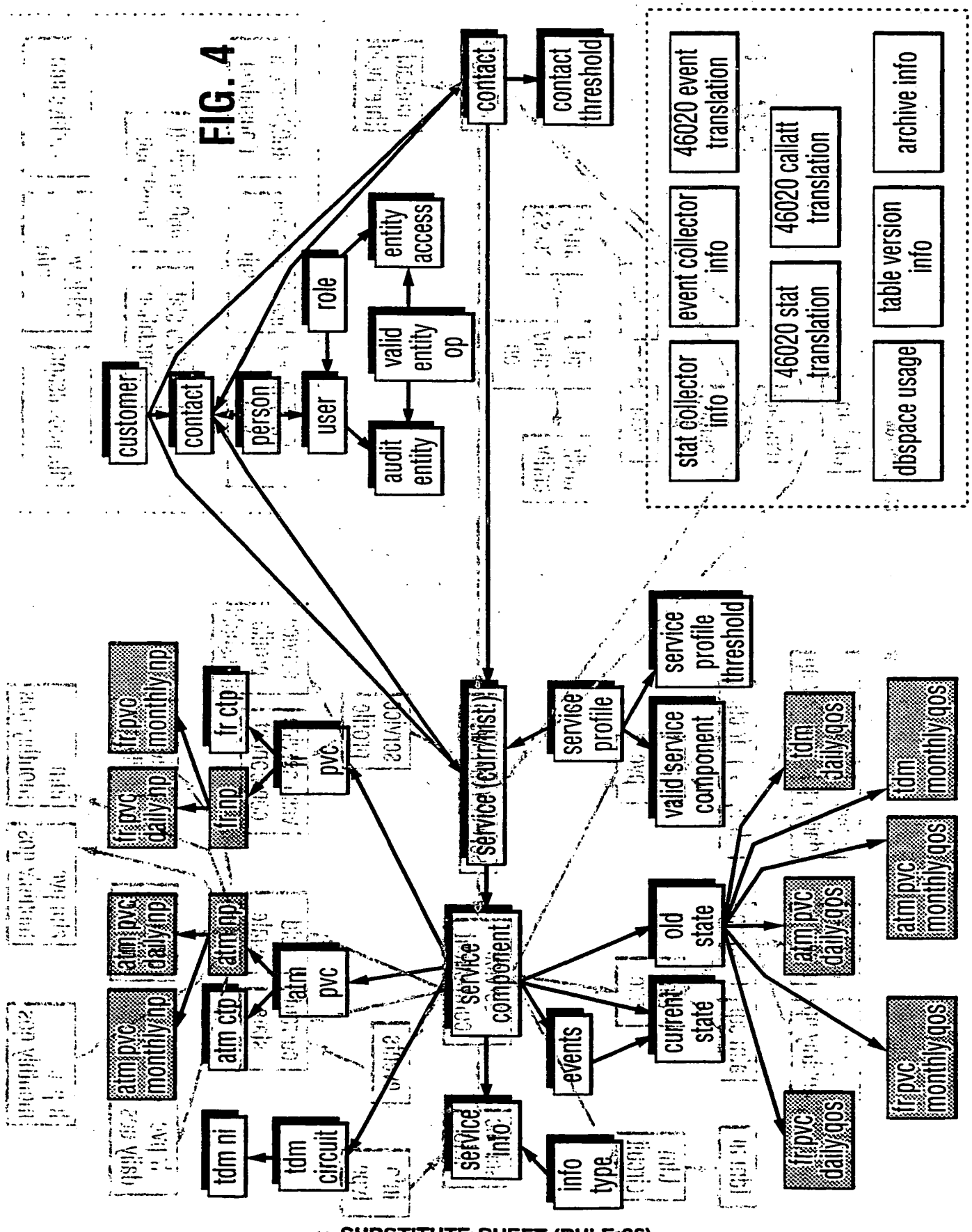


FIG. 5

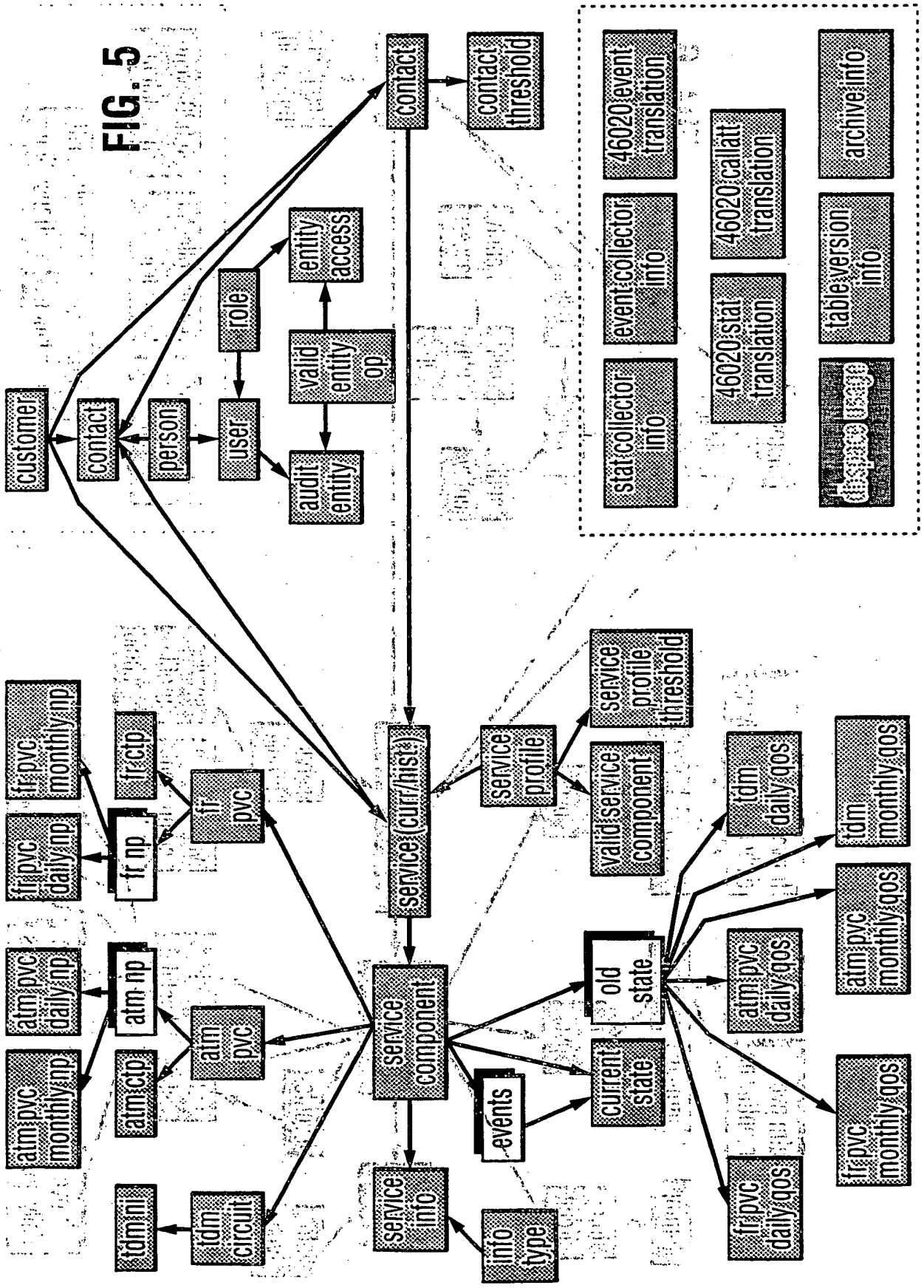


FIG. 6

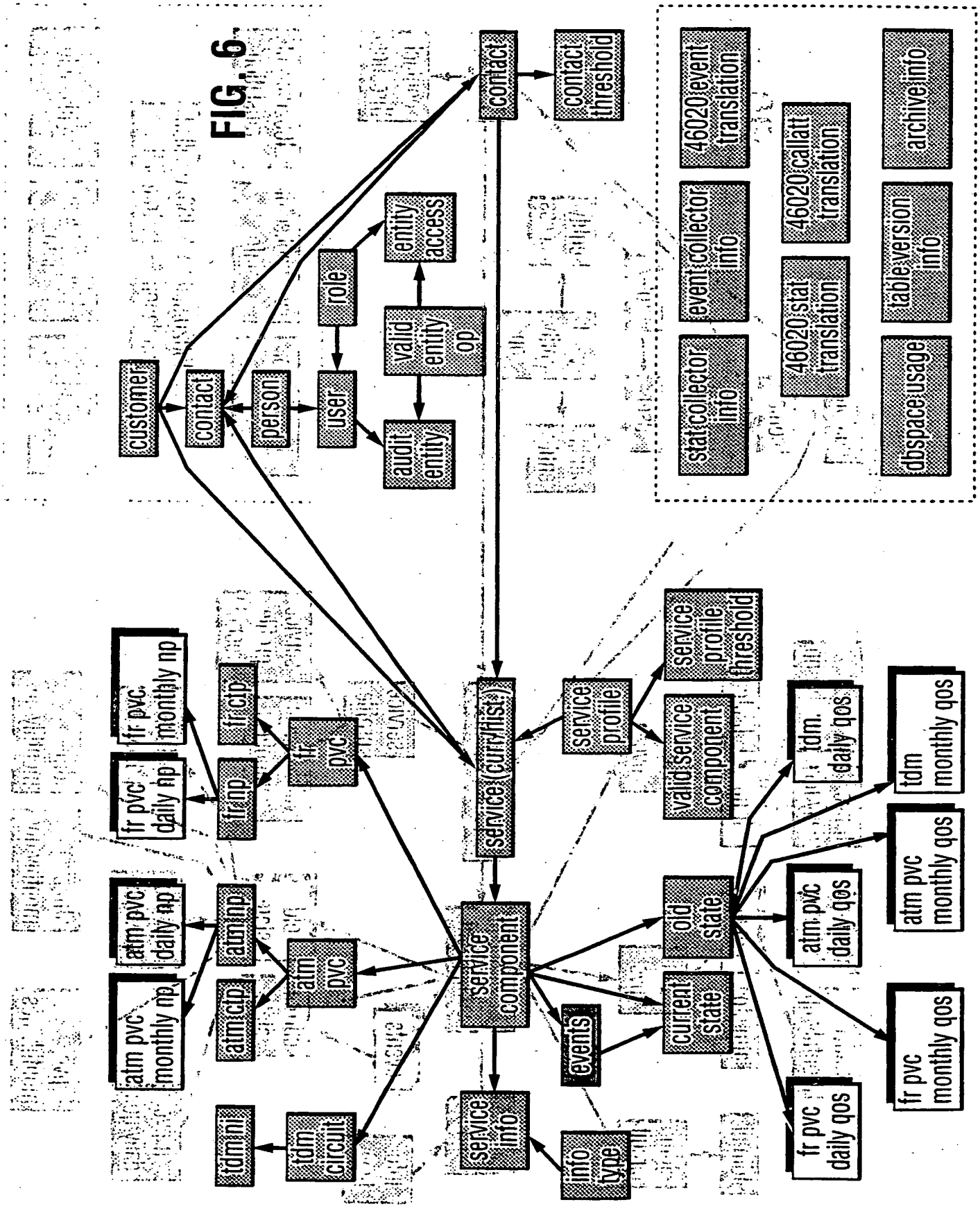
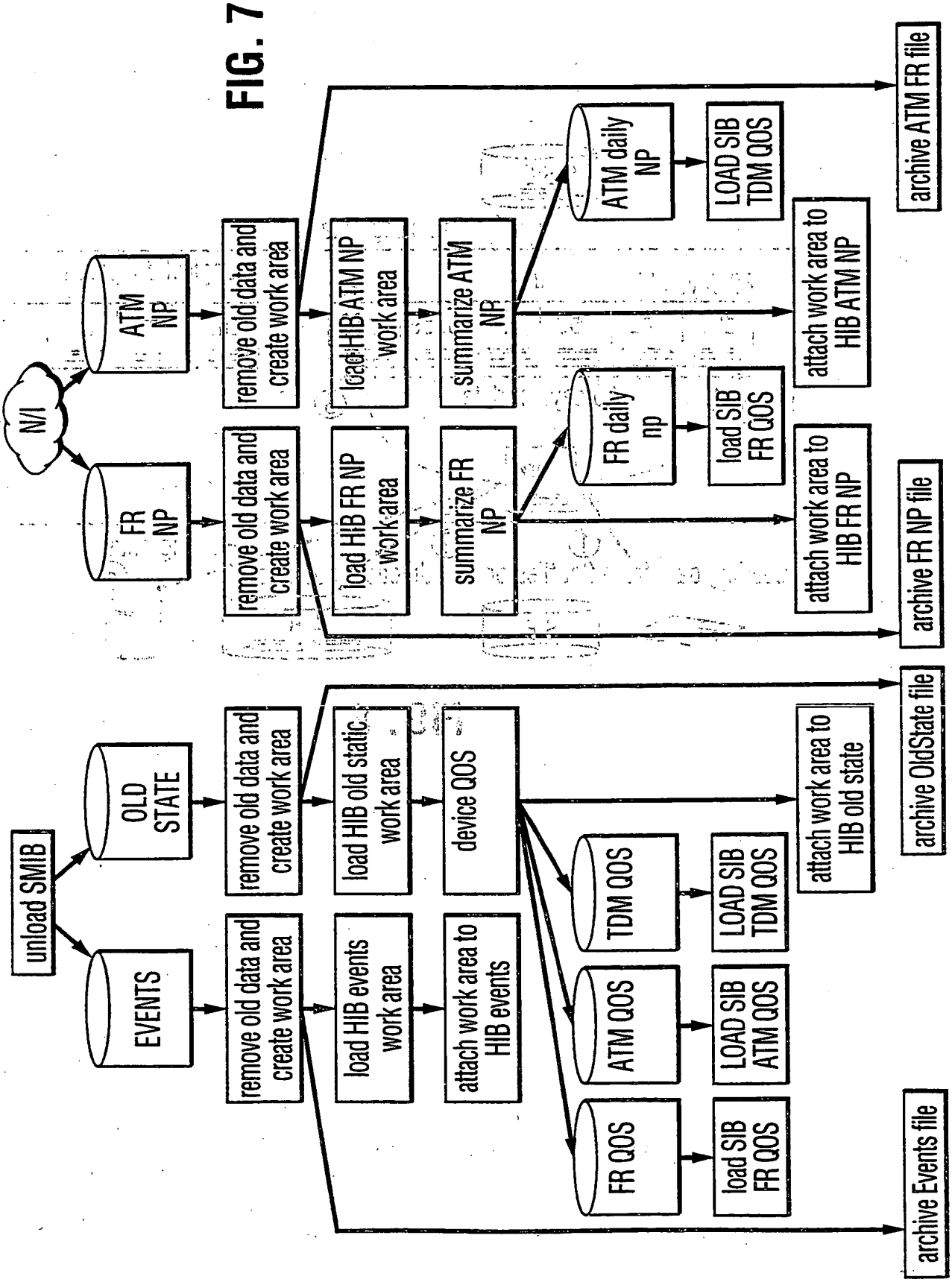


FIG. 7



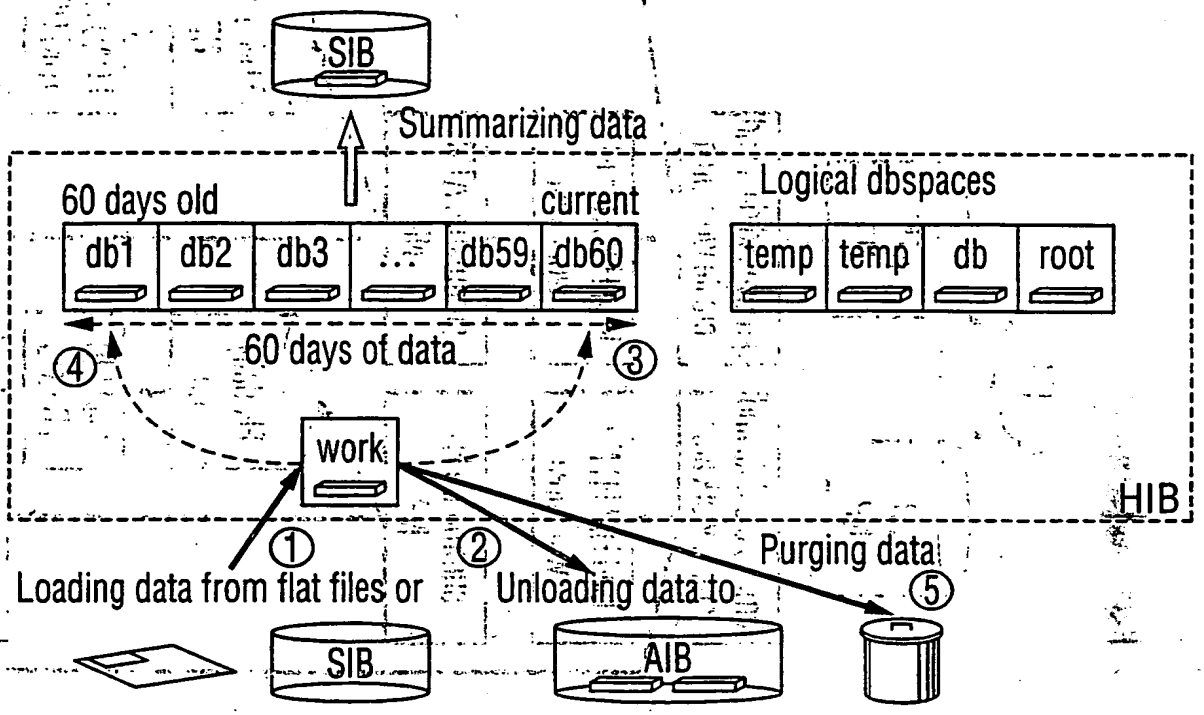
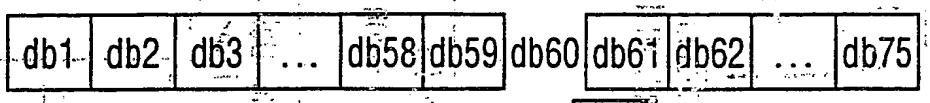


FIG. 8

DBSpace Usage Table

DBSpaceName	DBSpace Number	Object Type	Last Usage Date	Available
db1	6	event	1996-10-09	0
db2	10	event	1996-10-08	0
db3	14	event	1996-10-07	0
db4	18	event	1996-10-06	0
.	.	.	.	.
.	.	.	.	.
db60	262	event	1996-08-10	0
db61	266	event		1
db62	270	event		1

FIG. 9

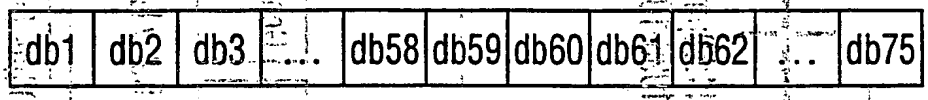


DBSpace Usage Table

db60

DBSpaceName	DBSpace Number	Object Type	Last Usage Date	Available
db1	6	event	1996-10-09	0
db2	10	event	1996-10-08	0
db3	14	event	1996-10-07	0
db4	18	event	1996-10-06	0
.	.	.	.	.
.	.	.	.	.
db60	262	event	1996-08-10	0
db61	266	event		1
db62	270	event		1

FIG. 10



DBSpace Usage Table

DBSpaceName	DBSpace Number	Object Type	Last Usage Date	Available
db1	6	event	1996-10-09	0
db2	10	event	1996-10-08	0
db3	14	event	1996-10-07	0
db4	18	event	1996-10-06	0
.	.	.	.	.
.	.	.	.	.
db60	262	event	1996-08-10	0
db61	266	event		1
db62	270	event		1

FIG. 11

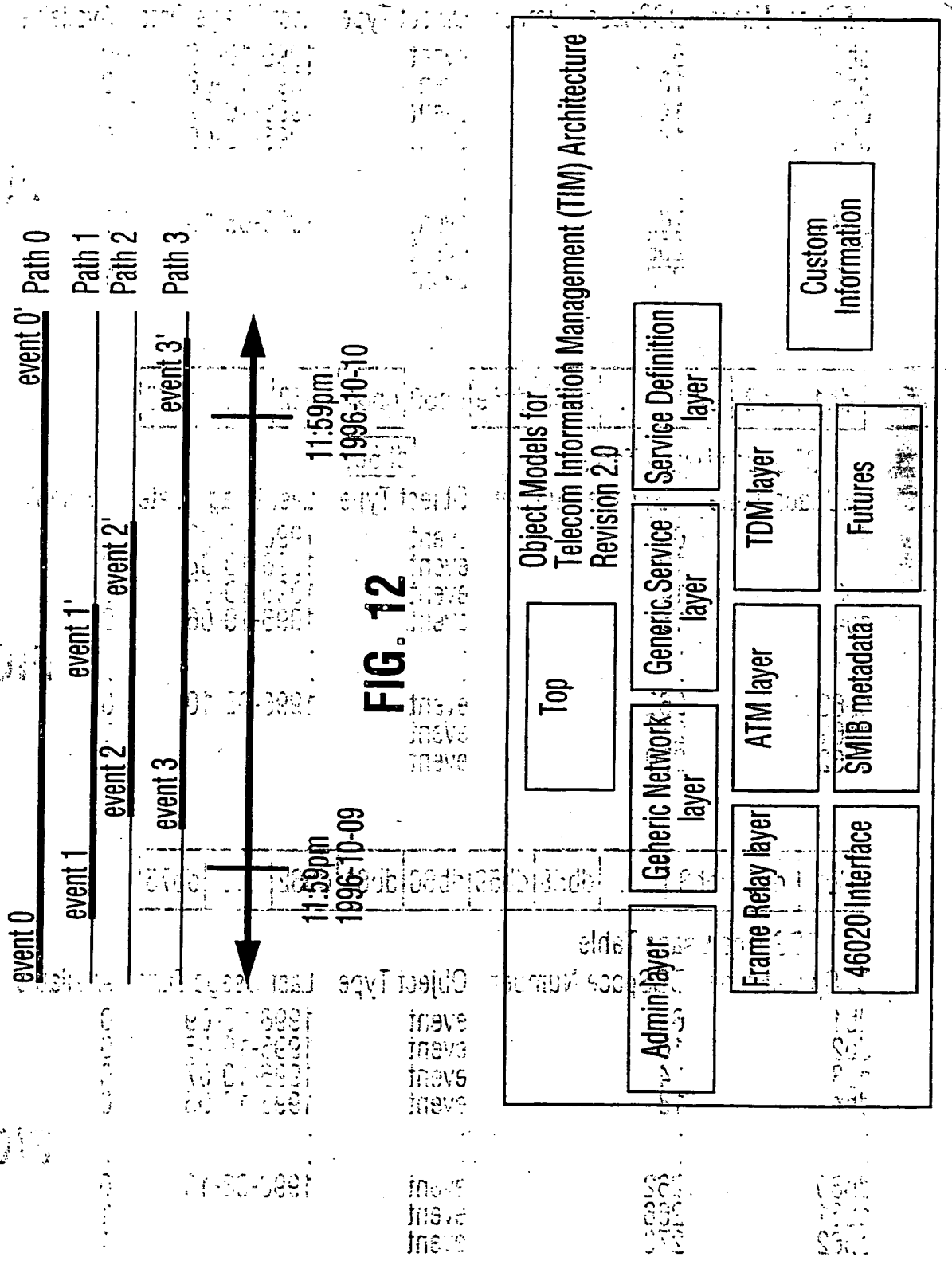


FIG. 12

FIG. 13

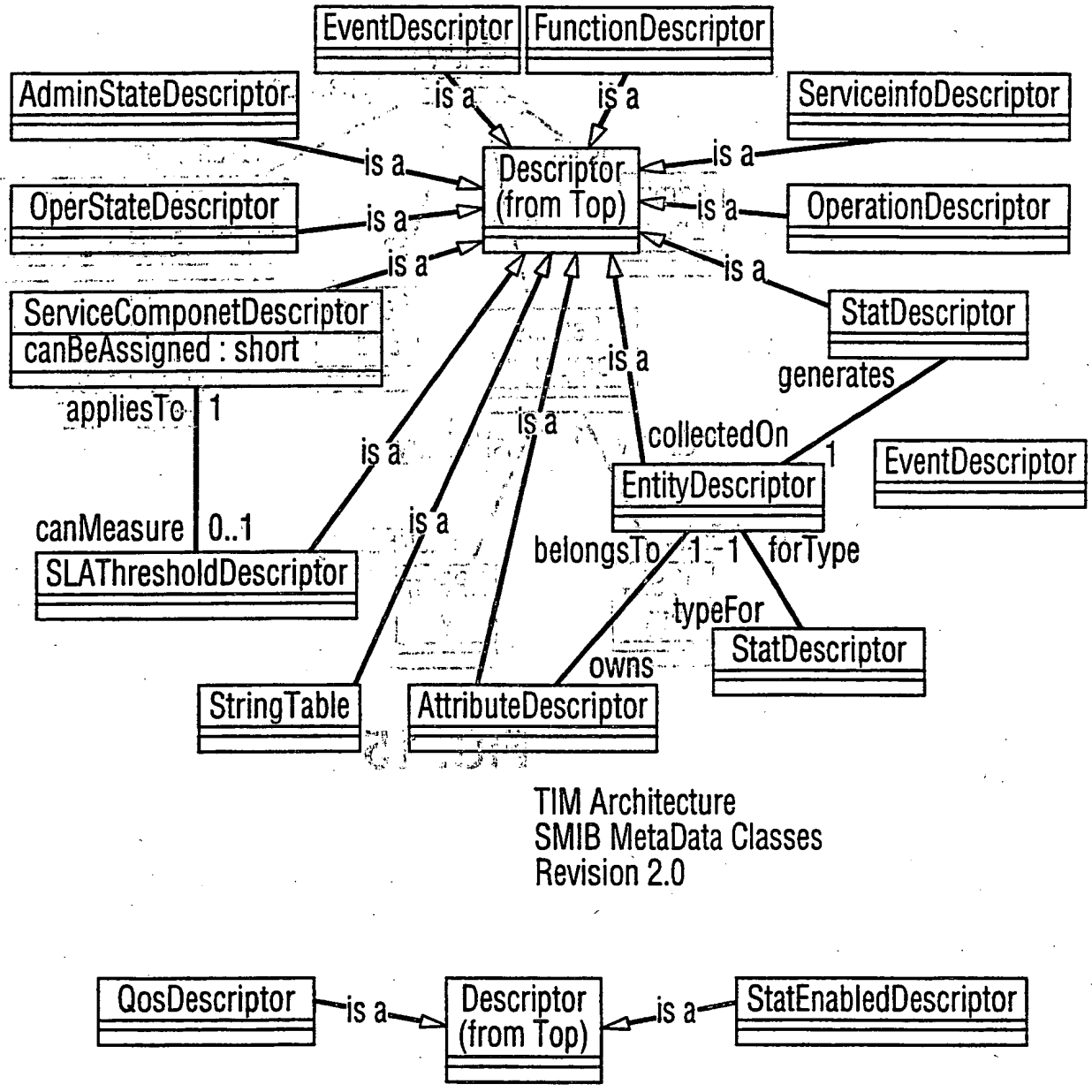


FIG. 14

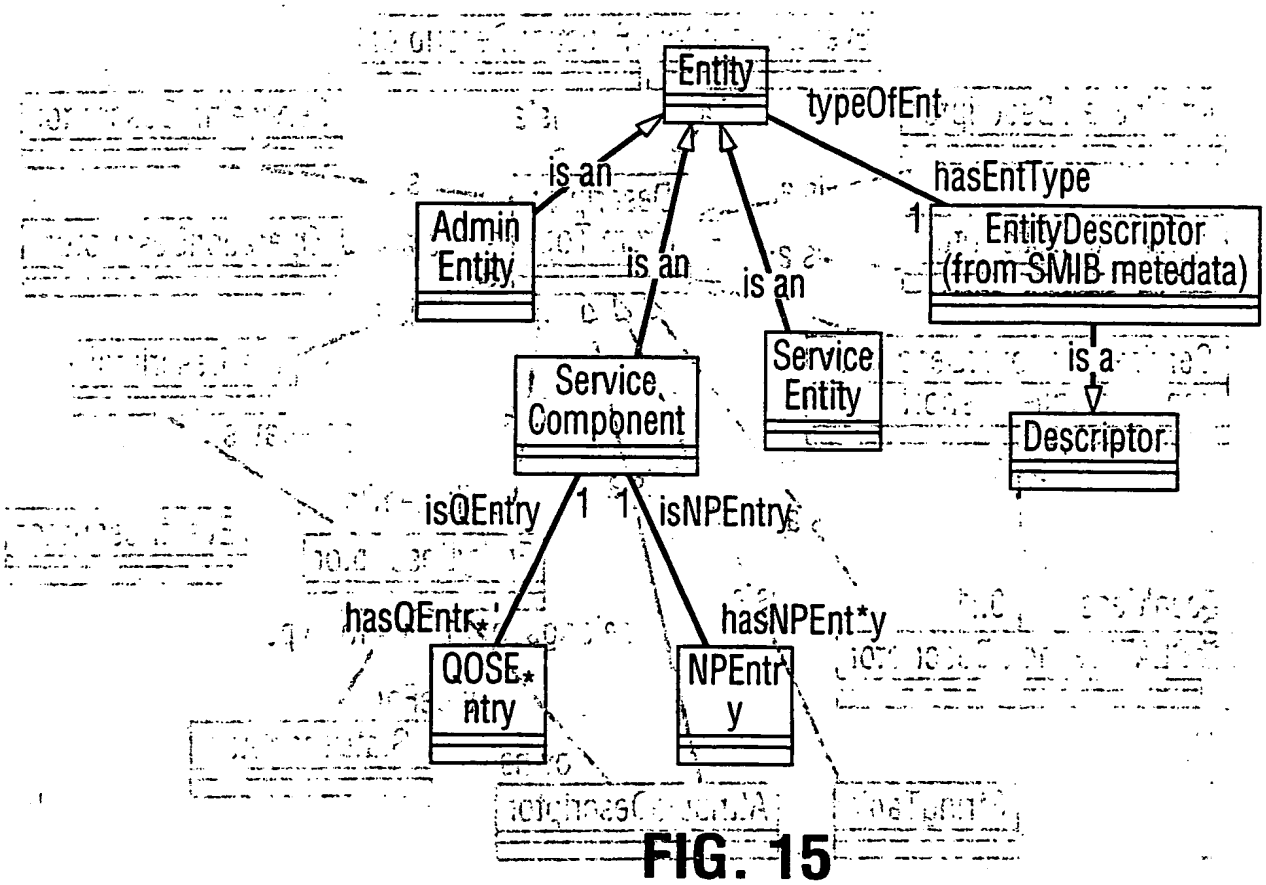


FIG. 15

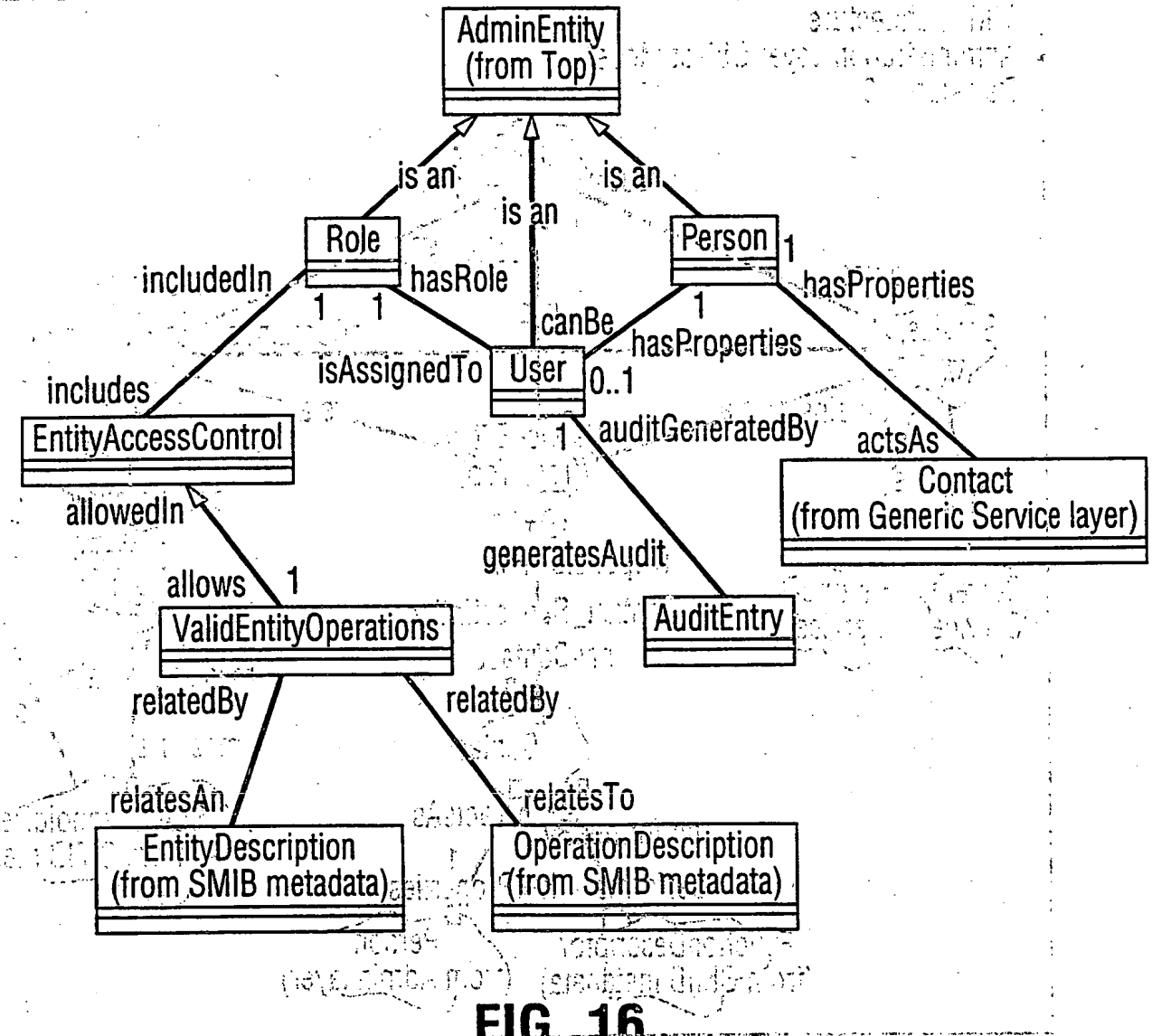


FIG. 16

TIM Architecture
Administration Layer Object Model
Revision 2.0

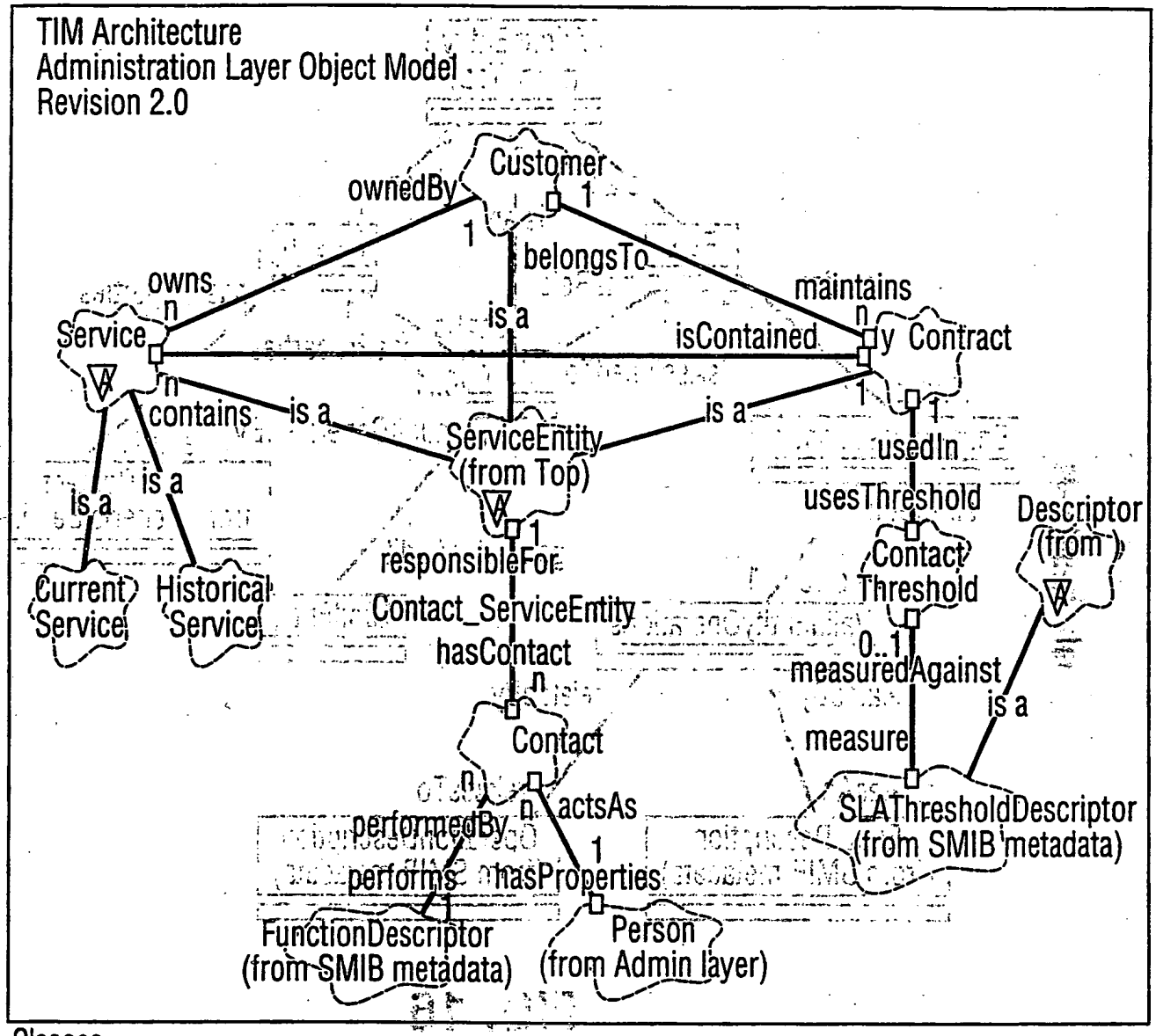


FIG. 17

INTERNATIONAL SEARCH REPORT

Intern ional Application No
PCT/CA 98/00232

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L12/24 H04Q11/04

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 96 07281 A (BRITISH TELECOMM ;AZARMI NADER (GB); CORLEY STEPHEN LESLIE (GB)) 7 March 1996 see page 6, line 7 - line 10 see page 14, line 18 - page 15, line 4 see page 30, line 14 - page 35, line 17	1,2,5-7, 9,11,12
A	WO 96 26588 A (CALBETRON SYSTEMS INC) 29 August 1996 see page 2, line 18 - page 2, line 28 see page 4, line 30 - page 5, line 6 see page 8, line 26 - page 9, line 1	1-14

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

' Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 July 1998	24/07/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Gregori, S

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Intern. ial Application No
PCT/CA 98/00232

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HINCHLIFFE M ET AL: "CUSTOMER NETWORK MANAGEMENT AND ATM NETWORKS" 1996 IEEE NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM (NOMS), KYOTO, APR. 15 - 19, 1996, vol. VOL. 1, no. SYMP. 5, 15 April 1996, INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, pages 155-164, XP000641088 see page 157, line 1 - line 8 see page 161, line 26 - page 162, line 18	1-14

Form PCT/ISA/210 (continuation of second sheet)(July 1992)

INTERNAT NAL SEARCH REPORT

Information on patent family members

Inter ial Application No
PCT/CA 98/00232

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9607281 A	07-03-1996	AU 3393595 A	22-03-1996
		CA 2198885 A	07-03-1996
		EP 0786187 A	30-07-1997
		FI 970849 A	25-04-1997
		JP 10504949 T	12-05-1998
		NO 970943 A	28-04-1997
		NZ 292213 A	26-05-1997
WO 9626588 A	29-08-1996	AU 5183796 A	11-09-1996
		EP 0811284 A	10-12-1997